

DIPLOMARBEIT

netCrawler

ausgeführt an der

Höheren Abteilung für Informationstechnologie

der Höheren Technischen Bundeslehranstalt Wien 3 Rennweg

im Schuljahr 2011/12

durch

Daniel Kern

Andreas Stefl

Thomas Taschauer

unter der Anleitung von

Prof.Dipl.-Ing. Clemens Kussbach

Wien, am 11.04.2012

KURZFASSUNG

Mit unserer Diplomarbeit netCrawler versuchen wir die Aufgaben, die uns als angehende Netzwerkadministratoren in den letzten Jahren im Unterricht an der HTL Rennweg immer wieder viel Zeit gekostet haben, zu erleichtern. Eine dieser Aufgaben ist die Visualisierung und Dokumentation eines Netzwerkes, welche wir sowohl mit einer grafischen, als auch einer Darstellung in Form einer Liste ermöglichen. Um diese Visualisierung zu bereitstellen zu können, wird anfangs das Netzwerk nach intermediären Geräten, unter anderem Router, Switch und Firewall, durchsucht.

Der Name “netCrawler” kommt von der Art und Weise wie Geräte erkannt werden. netCrawler kriecht sozusagen von Gerät zu Gerät, bis es das gesamte Netzwerk kennt und visualisieren kann. Es ist also ein “network crawler”, der “Netzwerk Kriecher”.

Weiters soll nicht nur das Netzwerk angezeigt werden, sondern auch mit wenig Aufwand jedes der dargestellten Geräte konfiguriert werden. netCrawler bietet dazu sowohl eine Vielzahl an vorkonfigurierten Werten an, als auch die Möglichkeit jeden beliebigen Wert auszulesen und zu verändern. Um möglichst viele Geräte und Szenarien zu unterstützen, ist es möglich die Geräte mithilfe mehrerer Protokolle, zum Beispiel SSH oder SNMP, zu konfigurieren.

Im Laufe der Diplomarbeit sind außerdem einige Nebenprodukte entstanden, die sowohl Netzwerkadministratoren, als auch Lehrer unterstützen.

ABSTRACT

With our final project netCrawler we aim to ease several time-consuming tasks we encountered in our experiences as network engineers in training within the last years at the HTL Rennweg. One of these tasks is the visualization and documentation of networks, which we accomplish using a graphical, as well as a visualization in form of a list. In order to build these visualizations netCrawler scans the network for intermediate devices, such as router, switch and firewall. Eventually, this information is being used to display the network.

The name “netCrawler” describes the method used for scanning the network. netCrawler crawls from device to device, until the whole network is known and can be displayed.

In addition to visualizing a network, it should also be possible to configure the displayed devices without too much hassle. netCrawler provides various preconfigured values for this purpose, but also allows users to read and configure whatever value they want. In order to support as much scenarios and devices as possible, we provide the ability to use various protocols, like SSH and SNMP, to configure devices.

During the course of this final project, a few spin-offs have been made available, which help network engineers and teachers.

EHRENWÖRTLICHE ERKLÄRUNG

Ich versichere,

- dass ich meinen Anteil an dieser Diplomarbeit selbstständig verfasst habe,
- dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe
- und mich auch sonst keiner unerlaubten Hilfe bzw. Hilfsmittel bedient habe.

Wien, am 11.04.2012

PRÄAMBEL

Die Inhalte dieser Diplomarbeit entsprechen den Qualitätsnormen für "Ingenieurprojekte" gemäß § 29 der Verordnung des Bundesministers für Unterricht und kulturelle Angelegenheiten über die Reife- und Diplomprüfung in den berufsbildenden höheren Schulen, BGBl. Nr. 847/1992, in der Fassung der Verordnungen BGBl. Nr. 269/1993, Nr. 467/1996 und BGBl. II Nr. 123/97.

Liste der betreuenden Lehrer:

- Prof. Dipl.-Ing. Clemens Kussbach
- Prof. Dipl.-Ing. Christian Schöndorfer
- Prof. Mag. Thomas Angerer

DANKSAGUNG

Wir möchten uns bei unseren Nebenbetreuern, Prof. Dipl.-Ing. Christian Schöndorfer und Prof. Mag. Thomas Angerer, bedanken, welche uns stets mit Rat und Tat zur Seite gestanden sind.

Insbesondere bedanken wir uns herzlichst bei unserem Hauptbetreuer, Prof. Dipl.-Ing. Clemens Kussbach, welcher uns immer motiviert hat, falls jemand Gefahr lief seine Begeisterung für die Diplomarbeit zu verlieren. Außerdem achtete er immer darauf, dass das Projekt auf Kurs blieb.

Des Weiteren möchten wir unseren Dank an die HTL Rennweg, den Abteilungsvorstand der Informationstechnologie, Mag. Dr. Gerhard Hager und Direktor Dipl.-Ing. Mag. Dr. Martin Weissenböck richten, da dieses Projekt ohne unserer Schule nicht zustande gekommen wäre.

Schlussendlich geht unser Dank noch an nachfolgend genannte Entwickler, die es uns durch die Freigabe ihrer Arbeit ermöglicht haben diese Diplomarbeit in diesem Ausmaß überhaupt erst umsetzen zu können:

- Google, mit google-gson für die De- und Serialisierung von Objekten
- cryptzone, mit MindTerm für die Implementierung von SSH
- Frank Fock und Jochen Katz, mit SNMP4J für die Implementierung von SNMP
- Andreas Stefl, mit common für die Erleichterung von häufig anfallenden Aufgaben
- sämtlichen freiwilligen und angestellten Entwicklern der nachfolgend erwähnten Technologien

INHALTSVERZEICHNIS

KURZFASSUNG.....	II
ABSTRACT.....	III
EHRENWÖRTLICHE ERKLÄRUNG.....	IV
PRÄAMBEL.....	V
DANKSAGUNG.....	VI
VORWORT.....	X
VERZEICHNIS DER TABELLEN.....	XI
VERZEICHNIS DER ABBILDUNGEN.....	XII
1 EINLEITUNG - TAS.....	15
1.1 IDEE.....	15
1.2 PROJEKTTEAM.....	15
1.3 MOTIVATION.....	15
2 PROJEKTMANAGEMENT - TAS.....	17
2.1 PLANUNG.....	17
2.2 PROJEKTKULTUR.....	20
2.3 ROLLENVERTEILUNG UND AUFGABEN.....	21
2.4 KONFLIKTE.....	22
3 MARKETING - TAS.....	25
3.1 SPONSOREN.....	25
3.2 WETTBEWERBE.....	26
3.3 AINAC.....	26
3.4 LIZENZIERUNG.....	26
4 ARBEITSWERKZEUGE - TAS.....	28

4.1	GOOGLE DOCS.....	28
4.2	GOOGLE GROUPS.....	31
4.3	GOOGLE TALK.....	32
4.4	GOOGLE SITES.....	33
4.5	JAVA.....	34
4.6	GIT UND GITHUB.....	35
4.7	ECLIPSE.....	39
4.8	BIBLIOTHEKEN.....	40
5	TESTWERKZEUGE - KER.....	42
5.1	PACKET TRACER TELNET BRIDGE - STE.....	42
5.2	CISCO SYSTEMS.....	51
6	TECHNOLOGIEN - KER.....	53
6.1	SNMP.....	53
6.2	LLDP.....	63
6.3	TELNET.....	65
6.4	SSH.....	68
6.5	CDP.....	71
7	UMSETZUNG - STE.....	73
7.1	VORAUSSETZUNGEN.....	73
7.2	PROGRAMMABLAUF.....	74
7.3	ERKENNUNG DER GERÄTE.....	75
7.4	AUSLESEN DER GERÄTEINFORMATIONEN.....	75
7.5	WICHTIGSTE GERÄTEINFORMATIONEN.....	77
7.6	SUCHLAUF.....	79

<u>7.7</u>	<u>INFORMATIONSVRWALTUNG.....</u>	<u>81</u>
<u>7.8</u>	<u>DARSTELLUNG DES NETZWERKES.....</u>	<u>92</u>
<u>7.9</u>	<u>DATENABLAG - TAS.....</u>	<u>92</u>
<u>7.10</u>	<u>SCHNITTSTELLE FÜR DRITTE - TAS.....</u>	<u>94</u>
<u>7.11</u>	<u>GRAFISCHE OBERFLÄCHE - TAS.....</u>	<u>95</u>
8	BENUTZERHANDBUCH - KER.....	97
<u>8.1</u>	<u>OBERFLÄCHE VON NETCRAWLER.....</u>	<u>97</u>
<u>8.2</u>	<u>ÄNDERN VON GERÄTEKONFIGURATIONEN.....</u>	<u>103</u>
9	ERGEBNIS - TAS.....	105
<u>9.1</u>	<u>NETCRAWLER.....</u>	<u>105</u>
<u>9.2</u>	<u>SCHOOLCRAWLER.....</u>	<u>106</u>
<u>9.3</u>	<u>PACKET TRACER BRIDGE.....</u>	<u>107</u>
<u>9.4</u>	<u>CONFIGURATION ASSISTANT.....</u>	<u>108</u>
10	FAZIT.....	110
<u>10.1</u>	<u>PERSÖNLICHE ERFAHRUNGEN.....</u>	<u>110</u>
11	AUSBLICK - TAS.....	113
<u>11.1</u>	<u>WIE GEHT ES MIT DEM PROJEKT WEITER?.....</u>	<u>113</u>
<u>11.2</u>	<u>GIBT ES INTERESSENTEN?.....</u>	<u>113</u>
12	QUELLENVERZEICHNIS.....	114

VORWORT

Dieses Buch ist ein Endergebnis unserer Diplomarbeit netCrawler, welches unsere Vorgangsweise, das Endprodukt, sowie das bei der Umsetzung erlangte Wissen beschreibt. Der Code des Programmes wurde veröffentlicht und ist unter folgender Adresse einsehbar: <https://github.com/TomTasche/netCrawler>

Zu Beginn dieser Diplomarbeit widmen wir uns den allgemeineren Themen, wie der Planung des Projektes, wie es überhaupt zu diesem Projekt gekommen ist und unserem Team.

Die darauffolgenden Kapitel bestehen aus technischen Grundlagen, Voraussetzungen und den verwendeten Technologien für das Programm. Es wurde versucht, möglichst technische, zugleich aber auch leicht verständliche Erklärungen zu finden.

VERZEICHNIS DER TABELLEN

TABELLE 1: DIE WICHTIGSTEN INFORMATIONEN EINES GERÄTES	
.....	79

VERZEICHNIS DER ABBILDUNGEN

ABBILDUNG 1: EIN KLEINER TEIL UNSERES ZEITPLANS.....	19
ABBILDUNG 2: DIE ROLLENVERTEILUNG IN UNSEREM PROJEKT.	21
ABBILDUNG 3: EIN AUSSCHNIT AUS UNSEREM GIT-GRAPHEN. . .	37
ABBILDUNG 4: DIE MULTIUSER-FUNKTION IN PACKET TRACER..	42
ABBILDUNG 5: MULTIUSER-CONNECTION IN PACKET TRACER ERSTELLT UND VERBUNDEN.....	43
ABBILDUNG 6: ABLAUF EINER PTMP-VERBINDUNG.....	45
ABBILDUNG 7: DER STACK DER BEI EINER MULTIUSER- VERBINDUNG DURCHLAUFEN WIRD.....	46
ABBILDUNG 8: ABLAUF EINER MULTIUSER-VERBINDUNG.....	47
ABBILDUNG 9: ÜBERBRÜCKUNG DES VIRTUELLEN NETZWERKVERKEHRS VON PACKET TRACER.....	48
ABBILDUNG 10: PACKET TRACER BRIDGE ALS SCHNITTSTELLE ZWISCHEN PACKET TRACER UND REALEN NETZWERKEN.....	49
ABBILDUNG 11: PACKET TRACER TELNET BRIDGE ALS SCHNITTSTELLE ZWISCHEN PACKET TRACER UND EINEM TELNET SERVER.....	50
ABBILDUNG 12: MÖGLICHE STRUKTUR EINES NETZWERKES VON SNMP-GERÄTEN.....	55
ABBILDUNG 13: ABLAUF EINER ABFRAGE ÜBER SNMP.....	59
ABBILDUNG 14: ABLAUF EINES LLDP-DIENSTES.....	64
ABBILDUNG 15: ABLAUF EINER TELNET-VERBINDUNG.....	65
ABBILDUNG 16: ABLAUF EINER SSH-VERBINDUNG.....	69
ABBILDUNG 17: ABLAUF EINES CDP-DIENSTES.....	72
ABBILDUNG 18: DER SUCHLAUF VISUALISIERT.....	80

ABBILDUNG 19: VISUALISIERUNG DER VERWENDETEN HIERARCHIE BEI DEN BASISKLASSEN.....	83
ABBILDUNG 20: VISUALISIERUNG DER VERWENDETEN HIERARCHIE BEI DEN ERWEITERUNGSKLASSEN.....	84
ABBILDUNG 21: DIE KLASSEN NETWORKMODEL, NETWORKMODELEXTENSION UND NETWORKMODELLISTENER IN EINER ÜBERSICHT.....	85
ABBILDUNG 22: DIE KASSEN TOPOLOGYDEVICE, TOPOLOGYDEVICELISTENER, TOPOLOGYINTERFACE UND TOPOLOGYCABLE IN EINER ÜBERSICHT.....	88
ABBILDUNG 23: DIE KLASSEN TOPOLOGY UND TOPOLOGYLISTENER IN EINER ÜBERSICHT.....	90
ABBILDUNG 24: DIE OBERFLÄCHE VON NETCRAWLER, WIE SIE NACH DEM STARTEN AUSSIEHT.....	97
ABBILDUNG 25: DIE OBERFLÄCHE VON NETCRAWLER, WIE SIE NACH EINEM SUCHLAUF AUSSIEHT.....	98
ABBILDUNG 26: DIE ZUVOR GECRAWLTE TOPOLOGIE IN FORM EINER LISTE.....	99
ABBILDUNG 27: DER BATCH MANAGER MIT AUTOMATISCH EINGETRAGENEN VERBINDUNGSDATEN.....	100
ABBILDUNG 28: DER BATCH MANAGER MIT EINER MÖGLICHEN BATCH UM DIE NETZWERKKONNEKTIVITÄT ZU KAPPEN.....	101
ABBILDUNG 29: DIE GERÄTE-ANSICHT.....	102
ABBILDUNG 30: DIE GERÄTE-ANSICHT IM EDITIER-MODUS.....	102
ABBILDUNG 31: DIE OBERFLÄCHE VON NETCRAWLER, WIE SIE NACH EINEM SUCHLAUF AUSSIEHT.....	106
ABBILDUNG 32: DIE AUSGABE EINES SUCHLAUFES MIT SCHOOLCRAWLER.....	107

**ABBILDUNG 33: DER CONFIGURATION MANAGER ZUM
ERSTELLEN VON KONFIGURATIONEN, DER CONFIGURATION
EXECUTOR ZUM AUSFÜHREN DIESER.....109**

1 EINLEITUNG - TAS

1.1 Idee

netCrawler entsprang einer Idee von Herrn Professor Schöndorfer im Zuge des Projektmanagement Unterrichts der vierten Klasse. Unser Team, welches damals aus vier Personen bestand, hatte die Aufgabe ein Programm zu entwickeln, welches ein Netzwerk erkennt, darstellt und vordefinierte Teile der Konfiguration anzeigt. Am Ende des vierten Jahrganges entschlossen sich drei der ursprünglich vier Personen dieses Projekt im Zuge einer Diplomarbeit weiterzuentwickeln und zu erweitern.

Wir haben uns jedoch entschlossen, dass der Programmcode aus dem Unterricht keine Grundlage für unsere Diplomarbeit sein soll. Somit standen wir bezüglich Programmierung wieder am Anfang. Trotzdem konnten wir die bereits gesammelte Erfahrung umsetzen. Diese Erfahrung ist nicht nur technischer Natur, sondern auch die Zusammenarbeit innerhalb und außerhalb des Teams.

1.2 Projektteam

Unser Team besteht aus Thomas Taschauer, Andreas Stefl und Daniel Kern. Thomas Taschauer und Andreas Stefl sind seit der ersten Klasse gemeinsam in der HTL. Daniel Kern ist in der dritten Klasse dazugekommen. Es wurde jedoch schnell klar, dass wir uns gut verstehen und somit stand einer Diplomarbeit nichts im Wege.

1.3 Motivation

1.3.1 Thomas Taschauer

Ich habe das Projekt nach Außen vertreten und war der Kontakt für Interessierte. Außerdem war ich für die Motivation des Teams und den Erfolg des Projekts zuständig.

Neben diesen organisatorischen Aufgaben war ich ebenfalls für Teile der Entwicklung zuständig.

1.3.2 Andreas Stefl

Mein Hauptaugenmerk galt der Entwicklung. Dazu gehörte die Erstellung der Programmstruktur, das Einplanen von etwaigen Fremdbibliotheken und die Umsetzung des endgültigen Programms.

Wichtig war auch die Entwicklung einer effizienten Auslesemethode, womit ich mich ebenfalls intensiv beschäftigt habe.

1.3.3 Daniel Kern

Ich habe mich in diesem Projekt um das Marketing gekümmert, also darum, dass die Leute von unserem Projekt erfahren.

Ich habe mich auch darum gekümmert, die notwendigen Informationen über verwendete Protokolle, wie SSH und SNMP, zu beschaffen, damit die Entwicklung reibungslos verläuft.

Außerdem war ich Hauptverantwortlicher für die technische Dokumentation unseres Produktes.

Des Weiteren habe ich die Aufgabe des Dokumentenverantwortlichen übernommen, welcher dafür sorgt, dass die Dokumente mit einheitlichen Namen versehen werden, und immer an den richtigen Orten zu finden sind. Außerdem müssen diese Dokumente jederzeit abrufbar und immer aktuell sein, denn jedes Projektmitglied soll immer auf die aktuellsten Versionen zugreifen können.

2 PROJEKTMANAGEMENT - TAS

2.1 Planung

Erst nach einigen Besprechungen und Recherchen beschlossen wir, dass wir unser Projektmanagement an den besten Aspekten verschiedener Methoden orientieren wollten. Eine davon ist die Agile Methode, deren Prinzip wie folgt lautet:

Es gibt möglichst viel Freiraum und daher möglichst wenige Regeln. Dadurch bleibt der gesamte Prozess sehr flexibel und man kann sich mehr auf das Wesentliche konzentrieren.

Aufgrund des engen Zeitrahmens und bestimmten, von der Schule vorgegeben, Randbedingungen und Terminen, erweiterten wir diese Methode jedoch mit Elementen anderer Methoden. So orientierten wir uns anfangs stark an den Richtlinien der PMA, Projektmanagement Austria, da für den Genehmigungsprozess für Diplomarbeiten viele der Dokumente benötigt werden, die bei PMA üblich sind. Genauer gesagt, erstellten wir die folgenden Dokumente:

- Objektstrukturplan, welcher, wie der Name schon sagt, einen groben Überblick über die im Laufe des Projektes zu erstellenden Objekte schaffen soll
- Umweltanalyse, welche sämtliche bei der weiteren Planung in Betracht zu ziehenden Einflüsse auf das Projekt grafisch darstellt
- Risikoanalyse, welche die negativen Einflüsse der Umweltanalyse übernimmt, deren Auswirkungsgrad und Eintrittswahrscheinlichkeit bestimmt, sowie Gegenmaßnahmen definiert
- Organigramm, welches Rollen und deren Hierarchie festlegt

- Meilensteinplan, welcher die wichtigsten Termine definiert, zu denen eine wichtige Leistung von dem Team erwartet wird

Auf Rat eines unserer Nebenbetreuer haben wir zusätzlich einen Zeitplan in Form einer Tabelle erstellt, welcher sämtliche Deadlines und Termine beinhaltet. Außerdem ist daraus leicht ersichtlich, wer für den Termin verantwortlich ist und wie der entsprechende aktuelle Status aussieht, also ob der Termin eingehalten wurde, oder eine Verzögerung vorliegt. Somit wussten sowohl Projektleiter und Mitarbeiter, als auch die Betreuer des Projektes jederzeit auf einen Blick Bescheid wer, wann und vor allem welche Arbeiten abzuliefern hatte.

	29.8. - 4.9.	5.9. - 11.9.	12.9. - 18.9.	19.9. - 25.9.	26.9. - 2.10.	3.10. - 9.10.
Taschauer Thomas						Konzept für GUI erstellt
						Prototyp fertig
		Diplomarbeitenantrag fertig				
		Entwicklungs-Workflow festgelegt				
	Sämtliche bereits bestehende Dokumente korrigiert und erweitert	Grobes Konzept, samt Inhalte, der Abschlusspräsentation erstellt		Planung abgeschlossen		
Kern Daniel						Konzept für GUI erstellt
						Prototyp fertig
		Ziele SMARTisieren [1]				
		Diplomarbeitenantrag fertig				
	Urlaub [2]	Grobes Konzept, samt Inhalte, der Abschlusspräsentation erstellt	Grobes Inhaltsverzeichnis für Diplomarbeitenbuch erstellt	Planung abgeschlossen		

Abbildung 1: Ein kleiner Teil unseres Zeitplans

Dieser Zeitplan ist im Grunde mit dem Projektstrukturplan zu vergleichen, den wir bereits im Laufe des Unterrichts als eines der wichtigsten Dokumente in der von PMA definierten Richtlinien kennengelernt hatten. Der markanteste Unterschied zwischen den beiden Dokumenten liegt im Umfang der darin enthaltenen Informationen, da der Projektstrukturplan zusätzlich zu Endtermin, Verantwortlichen und zu erledigender Arbeit auch noch viele andere Informationen enthält, wie zum Beispiel einen Starttermin, Vorgänger, Nachfolger und vieles mehr. Genau hier setzt bei uns die Agile Methode ein: dem für eine Arbeit Verantwortlichen wird möglichst viel Freiraum geboten, indem, außer einem Endtermin, keine Vorschriften gemacht

werden. Soweit es die Abhängigkeiten einer Arbeit, sprich wichtige andere Arbeiten, erlauben, könnte ein Mitarbeiter bereits Monate vor dem Endtermin mit dieser Tätigkeit beginnen, wenn er das wollte.

Eine so freie Projektmanagementmethode einzusetzen birgt natürlich Probleme, welche wir aber mittels Vertrauen und ständiger Motivation vermeiden konnten. Eine mögliche negative Folge unserer Methode hätte sein können, dass Arbeiten, aufgrund der fehlenden Starttermine, erst kurz vor dem Endtermin begonnen werden und somit nicht die nötige Qualität aufweisen. Durch ständige Absprache und Motivation zwischen Mitarbeitern und Betreuern wurde bei uns jede Arbeit rechtzeitig begonnen und mit der gewünschten Qualität abgeliefert.

Ergänzend zu dem Zeitplan haben wir eine ToDo-Liste geführt, welche eine scheinbar chaotische Anordnung von anstehenden Arbeiten war. Sie war als eine leichter lesbare und laufend aktualisierte Darstellungsform des Zeitplans anzusehen und wurde ebenfalls dafür genutzt Arbeitsnotizen machen zu können.

2.2 Projektkultur

Für unser Projekt und jegliche dabei anfallende Kommunikation haben wir uns dazu entschieden, stets eine unverkrampfte Atmosphäre zu schaffen. Dazu war es uns wichtig auch formlose E-Mails zu akzeptieren. Hier war es einzig und allein entscheidend den Betreff entsprechend anzupassen, indem man ihn mit "netCrawler" versieht, und, wenn passend, die Mail ebenfalls an unseren E-Mail-Verteiler zu senden.

Ein weiterer wichtiger Baustein unserer Projektkultur war es, jegliche Konflikte innerhalb und außerhalb des Teams sofort mit allen anderen Mitarbeitern zu besprechen, anstatt verdeckte Konflikte sich ausbreiten

zu lassen oder Mitarbeiter bei außenstehenden Personen anzuschwärzen. Nachdem ein Konflikt intern besprochen wurde, wurde er je nach Dringlichkeit bei einem extra dafür angelegten, oder dem nächsten geplanten Meeting noch einmal mit dem Hauptbetreuer und gegebenenfalls den Nebenbetreuern besprochen. Somit konnte ebenfalls eine Vertrauensbasis zwischen Mitarbeitern untereinander, und zwischen Mitarbeitern und Betreuern hergestellt werden, da niemand besorgt sein musste, dass ihm etwas nicht mitgeteilt würde.

2.3 Rollenverteilung und Aufgaben

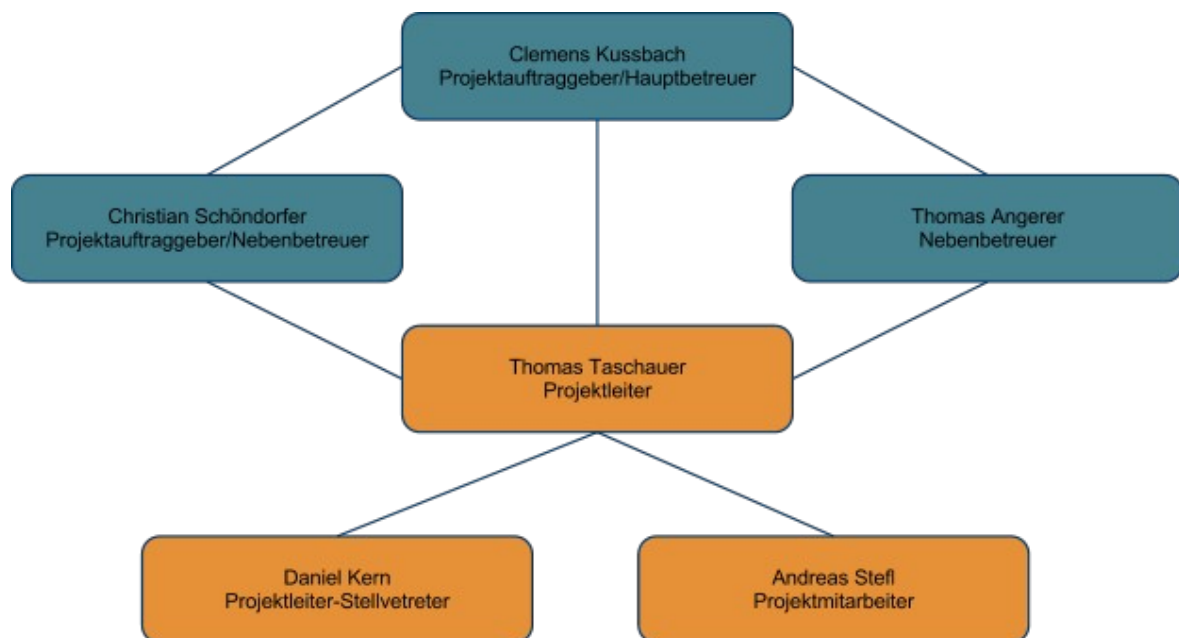


Abbildung 2: Die Rollenverteilung in unserem Projekt

Da es in einem Projekt sehr wichtig ist, dass die Rollen und Aufgaben klar definiert sind, haben wir uns zu Beginn unserer Diplomarbeit überlegt, wer für welche Spezialgebiete am besten geeignet sei. Aufgrund der verschiedenen Kompetenzen entschieden wir uns für folgende Einteilung:

Thomas Taschauer wurde zu unserem Projektleiter. Er vertrat netCrawler nach außen und organisierte die Aufgaben innerhalb des Projektes. Thomas war sehr geeignet für diese Position, da er sehr ordentlich ist, gewissenhaft arbeitet und ein kompetenter Techniker ist. Aus früheren Schulprojekten, in denen er ebenfalls als Projektleiter fungierte, brachte er außerdem Wissen mit, dass uns erlaubte Ziele schneller umsetzen zu können.

Andreas Stefl war unser Programmierer und somit hauptsächlich für die Entwicklung des Programmes zuständig. Er war sehr geeignet für diese Aufgabe, da er bereits vor seiner Laufbahn an der HTL damit begonnen hatte und sich somit entsprechend gut in diesem Themengebiet auskannte.

Daniel Kern war der stellvertretende Projektleiter, sowie Zuständiger für die Dokumentation der Arbeiten. Er vertrat Thomas bei Ausfällen und half ihm bei der Organisation, Planung und dem Marketing. Sein Hauptaugenmerk lag jedoch auf der Dokumentation des Geschehenen, sowie der Erkundung verschiedenen Technologien und Protokolle. Außerdem übernahm er die Entwicklung der grafischen Oberfläche für netCrawler.

2.4 Konflikte

2.4.1 Konflikte innerhalb des Teams

Im Laufe unserer Diplomarbeit gab es keine ernsthaften Konflikte innerhalb des Teams, was wahrscheinlich darauf zurückzuführen ist, dass wir schon in mehreren Projekten als Team zusammen aufgetreten sind. Somit wusste jeder von Anfang an über die Stärken und Schwächen des anderen Bescheid und konnte sein Verhalten dementsprechend anpassen. Diese Tatsache ersparte uns auch ein anfängliches "Ausloten" der Grenzen und Hierarchie, welches sonst in

jedem neuen Team früher oder später anfällt. In unserem Fall ist das schon in den vergangenen Schuljahren und Projekten geschehen, wodurch die daraus gewonnene Zeit in die Planung und Umsetzung investiert werden konnte.

Konfliktpotential herrschte jedoch immer wieder, wenn es darum ging Aufträge zu erteilen und Anweisungen zu geben. Die Schwierigkeit dabei liegt generell darin, dass sowohl Auftraggeber als auch Auftragnehmer wissen müssen, wie mit der Anweisung umzugehen ist. Ist der Auftrag für den Auftragnehmer nicht zufriedenstellend, weil er zu schwer, zu leicht, oder in zu kurzer Zeit zu erledigen ist, oder der Auftraggeber einfach die falschen Worte oder einen falschen Ton bei der Erteilung der Aufgabe verwendet, kann das bei schlechter Kommunikation in einem ernsthaften Konflikt ausarten. In unserem Fall war es aber weniger dieses Problem, sondern die Tatsache, dass, abgesehen von der Projekthierarchie wie sie auf dem Papier steht, sämtliche Mitglieder des Teams auf ein und der selben Ebene standen. Wenn man von Rollen wie "Projektleiter", "Stellvertretender Projektleiter" und "Projektmitarbeiter" absieht, haben hier drei, sozusagen "gleichrangige", Schüler an einem Projekt gearbeitet. Das machte es sehr schwer die geplante Projekthierarchie, speziell die Rolle des Projektleiters, in die Realität umzusetzen und den, eigentlich gleichrangigen, Projektmitarbeitern Aufträge und Anweisungen zu erteilen. In einem Unternehmen wird die Umsetzung solcher Projekthierarchien durch Faktoren wie das Einkommen einer Person, Druck auf Mitarbeiter in Form von Einfluss auf deren Gehalt und Drohung mit Kündigung durchgesetzt. Wir mussten hier jedoch rein mit Diskussionen und Einigungen auskommen.

2.4.2 Konflikte außerhalb des Teams

Bei der Zusammenarbeit mit Personen außerhalb des Teams gab es während der Diplomarbeit ebenfalls keine groben Auseinandersetzungen. Uns ist jedoch klar geworden, wie wichtig eine

gründliche Umweltanalyse in der Planungsphase ist. Nur wenn diese gut durchdacht und ausgearbeitet wurde, kann im späteren Verlauf des Projektes auf sie Rücksicht genommen werden und entsprechende Maßnahmen gesetzt werden. Diese Maßnahmen variieren je nach Art des Einflusses und können von einer Anpassung der Projektkultur im Hinblick auf Wünsche eines Betreuers, bis zur Wahl der Betreuer reichen. Konkret in unserem Projekt versuchten wir unseren Projektmanagement-Stil an die Wünsche mehrerer dem Team außenstehenden Einflüssen anzupassen, indem wir uns näher an den PMA-Richtlinien orientierten. Weiters haben wir stets relevante Neuigkeiten an Interessierte weitergegeben, um deren Interesse und Mitwirken an unserem Projekt aufrecht zu erhalten.

Einer unserer wichtigsten, und auch positivsten, Einflüsse in diesem Projekt war unser Hauptbetreuer, mit dem es im Laufe des Projektes keine Konflikte gab. Da dies jedoch die erste Diplomarbeit war, die unser Hauptbetreuer betreut hatte, fehlten ihm einige der Erfahrungen, die andere Betreuer bereits gesammelt hatten. Diese Erfahrungen reichen von voraussichtlichen Terminen bis zu Normen, denen Diplomarbeiten entsprechen müssen. Dieses von Anfang klar kommunizierte Defizit konnte aber durch Absprache mit anderen Betreuern und ehemaligen Diplomarbeitsteams wettgemacht werden.

3 MARKETING - TAS

Ein Problem dem wir uns beim Marketing stellen mussten, war die Frage wann wir mit dem Marketing, also der Suche nach Sponsoren und Wettbewerben, beginnen sollten. Einerseits haben Firmen nur ein begrenztes Budget, das für Sponsoring zur Verfügung steht und daher schon früh ausgeschöpft ist. Andererseits hatten wir Anfangs noch keinen Prototypen vorzuweisen, der unseren Vorstellungen entsprochen hat. Trotzdem haben wir schon im Dezember damit begonnen Sponsoren zu kontaktieren. Dabei ging es uns grundsätzlich aber weniger um das Geld, sondern um Feedback zu unserer Arbeit und einer Zusammenarbeit mit den Firmen. Am wichtigsten war uns dabei eine Rückmeldung zu unseren geplanten Funktionen für netCrawler zu bekommen, da wir als Schüler noch nicht genau sagen konnten, was für einen realen Netzwerkadministrator entscheidend ist.

3.1 Sponsoren

Zusammen mit unseren Betreuern haben wir einige Unternehmen ausgewählt, die an unserem Projekt sein hätten können. Bei einigen Firmen, wie T-Systems, Kapsch, Schrack, UPC und PCS-IT kam es nie über Telefonate hinaus, trotzdem konnten wir mit einer Firma, nämlich der KSI, Kontakt-Systeme Inter, genauer ins Gespräch kommen.

3.1.1 KSI

Die KSI war nach einem ersten Telefonat spontan bereit sich mit uns zu treffen um genaueres über unser Vorhaben zu erfahren. Bei der Besprechung stellte sich letztendlich heraus, dass sie für unsere Projekte keine Verwendung finden würden, da sie ausschließlich Netzwerkbestandteile verkauft, jedoch selbst keine Netzwerke verwaltet.

Eine Funktion die sie sich gewünscht hätten, nämlich die Überwachung der Performance und des Status eines Netzwerkes, konnten wir im Rahmen dieser Diplomarbeit nicht einbinden, da wir genau dieses Szenario in einem Nicht-Ziel verfasst hatten.

Am Ende der Besprechung, wurden wir von der KSI noch an eine Firma namens Rabcat vermittelt. Diese ist eigentlich zuständig für Spiele und Grafiken, hat aber vor kurzem von der KSI ein Netzwerk bekommen, weshalb netCrawler für sie interessant sein könnte. Aufgrund eines vollen Terminkalenders bei dieser Firma konnte noch kein intensiver Kontakt hergestellt werden.

3.2 Wettbewerbe

Leider kam für uns keiner der uns bekannten Wettbewerbe in Frage, da meistens auf künstlerische Arbeit Wert gelegt wurde.

3.3 AINAC

Am 14.03.2012 hatten wir im Zuge der AINAC, der Austrian International Network Academy Conference, die Möglichkeit unser Projekt einem breiteren Publikum vorzustellen. Trotz eines kleineren Publikums als erwartet, war diese Präsentation für uns eine gute Erfahrung. Das Publikum hat einige Fragen gestellt und uns somit gezeigt, wo die Stärken und Schwächen unseres Projektes liegen. Außerdem konnten wir erfolgreich alle unsere Produkte live ohne Probleme demonstrieren, weshalb wir diese Art der Präsentation auch bei zukünftigen Auftritten anwenden werden.

3.4 Lizenzierung

Wir haben uns schon sehr früh dazu entschieden unsere gesamte Arbeit als Open-Source zu veröffentlichen, um somit Interessierten zu ermöglichen netCrawler ohne Einschränkungen und Bedenken einzusetzen. Das Problem bei der Wahl einer passenden Open-Source

Lizenz ist zu entscheiden, welche Bedingungen für Dritte bestehen sollen, wenn sie den entsprechenden Code verwenden oder modifizieren. Einerseits gibt es Lizenzen, die eine Veröffentlichungen jeglicher Änderungen an dem ursprünglichen Code erzwingen, andererseits gibt es auch jene, die keine solchen Maßnahmen verlangen. Um ein Open-Source Projekt lebendig zu halten, ist es wichtig Änderungen von anderen wieder zurück zu bekommen, um diese in das Projekt einzuspeisen. Andererseits ist es aber auch wichtig das eigene Projekt für Firmen und andere Entwickler interessant zu machen, in dem man ihnen wenige Vorschriften macht.

Für dieses Projekt haben wir uns entschieden unsere Arbeit unter der LGPL, der Lesser General Public License, einer abgeschwächten Version der sehr verbreiteten und umstrittenen GPL, zu veröffentlichen, da diese eine Mischung aus Vorteilen für uns und Vorteilen für interessierte Firmen und Entwickler bietet. Genauer gesagt, erlaubt die LGPL Dritten unser Programm in kompilierter Form, also zum Beispiel eine ausführbare Datei, ohne Einschränkungen zu verwenden. Sobald aber Änderungen an unserem Code vorgenommen werden, müssen diese wiederum unter der LGPL als Open-Source Code zugänglich gemacht werden.

4 ARBEITSWERKZEUGE - TAS

4.1 Google Docs

Wenn man vor der Entscheidung steht welches Textverarbeitungsprogramm man verwenden will, spielen viele Faktoren eine wichtige Rolle. Außerdem beeinflusst diese Wahl ebenfalls einige weitere wichtige Beschlüsse, die sowohl die Projektkultur, als auch den Arbeitsablauf betreffen. Daher sollte man vor der endgültigen Wahl des Textverarbeitungsprogrammes einige Aspekte in Betracht ziehen. Eine der gravierendsten Entscheidungen ist die Wahl des Formates, in dem Dokumente abgespeichert werden sollen. Die größten Vertreter sind hierbei die Familie des proprietären Word-Formates von Microsoft und die freien Standards der Open Document Formats, kurz ODF. Aufgrund der weiten Verbreitung, und manchmal auch aufgrund fehlender Kenntnis über Alternativen, werden Microsofts Word-Formate von Firmen bevorzugt. Diese Wahl hat zur Folge, dass Dokumente auch nach außen von sehr vielen Kommunikationspartnern geöffnet und betrachtet werden können, da sowohl Konkurrenzprodukte, als auch Microsofts eigene weit verbreitete Produkte auf dieses Format setzen. Der Nachteil wiederum ist, dass aufgrund mangelnder Standardisierung von Seiten Microsofts, jegliche Parteien ohne Zugriff auf Microsoft Produkte nur eine Annäherung an das originale Dokument betrachten können. Solch eine Partei vorzufinden wird in Zukunft immer öfter der Fall sein, nachdem sich bereits einige Gemeinden, Städte und sogar Firmen entschlossen haben die verlangten Lizenzkosten für den Einsatz von Microsoft Produkten einzusparen, indem sie auf freie Produkte wie LibreOffice und OpenOffice umsteigen. Um solche und ähnliche Kompatibilitätsprobleme zu vermeiden ist es ratsam Dokumente an außenstehende ausschließlich in anderen weit verbreiteten Formaten, wie dem PDF, Portable Document Format, oder auch HTML, Hypertext Markup Language, zu versenden. Diese Formate verhindern außerdem irrtümliche Veränderungen an dem Dokument.

Ein weiterer wichtiger Faktor für die oben genannte Entscheidung ist die Art der so genannten Dokumentenablage, also der Art und Weise, wie und wo Dokumente abgespeichert werden. Die Definition der Dokumentenablage umfasst die Häufigkeit der Sicherung, reichend von sofortiger Sicherung bis zu einer Generalsicherung jeden Tag, bis zu dem Namen der einzelnen Dokumente. Legt man also zum Beispiel fest sämtliche Dokumente erst nach jeder Änderung zu sichern, wäre es ratsam eine Art Versionsnummer darin festzuhalten. Entscheidet man sich jedoch für eine Umgebung, in der sämtliche Änderungen sofort, ohne Zutun des Nutzers gesichert werden, ist eine solche Versionsnummer meist eher unpassend.

Weitere zu bedenkende Einflüsse, sind sowohl die Ausrüstung der Mitarbeiter, deren Kenntnisse, als auch die Kosten- und Sicherheitsansprüche der Firma. Bei hochsensiblen Daten sollte zuerst überprüft werden, ob außenstehende Anbieter auch die nötige Sicherheit gewährleisten können. Bei einer lokalen Selbstinstallation der Dienste sind aber auch die Kosten und damit verbundener Aufwand nicht zu unterschätzen.

Für uns spielten all diese Faktoren eine Rolle, weshalb uns, im Angesicht der anstehenden Dokumentationsarbeit, eine gut bedachte Entscheidung sehr am Herzen lag. In unserem Fall erleichterten uns gewisse Rahmenbedingungen die Entscheidung, wie zum Beispiel ein Mangel an monetären Mitteln für teure Lizenzen und die Vorkenntnisse sämtlicher Mitarbeiter. Somit fiel die Entscheidung einstimmig auf das "Text & Tabellen"-Angebot von Google, genannt Google Docs. Dieses steht kleinen Firmen unter 10 Mitarbeitern ohne jegliche Einschränkungen kostenlos zur Verfügung. Einzig an menschlicher Hilfe bei Problemen mangelt es diesem Angebot, was aber aufgrund der

vergleichsweise kurzen Dauer unseres Projektes kein Problem darstellte. Ein weiterer Kritikpunkt, der anfangs im Raum stand, ist die fehlende Erreichbarkeit von Dokumenten, wenn einmal kein Internetzugriff vorhanden ist, da Google Docs die Dokumente ausschließlich im Internet abspeichert, anstatt sie, wie von herkömmlichen Produkten gewohnt, lokal am Computer des Nutzers zu speichern. Aufgrund der drastisch ansteigenden Möglichkeiten um jederzeit an jedem Ort mit dem Internet verbunden sein zu können, legten wir auch diese Sorge ab. Bei einem voraussehbaren Mangel an Konnektivität bot uns Google Docs die Möglichkeit Dokumente im Voraus in einer Vielzahl verschiedener Formate herunterzuladen, zu editieren und später wieder in das gewohnte Umfeld zu importieren. Somit war es uns in beinahe allen Fällen möglich an unserer Diplomarbeit weiterzuarbeiten.

Erst während der aktiven Nutzung der angebotenen Produkte wurde uns klar, dass Google Docs noch viel mehr bietet, als es anfangs scheinen mochte. So lernten wir besonders die Möglichkeit zu schätzen, ein und das selbe Dokument mit mehreren Mitarbeitern gleichzeitig zu bearbeiten. Normalerweise würde der Arbeitsablauf in etwa so aussehen, dass man sich die aktuellste Version eines Dokumentes herunterladen würde, dieses bearbeitet und anschließend wieder an den entsprechenden Ort ablegt. Das hat zur Folge, dass mühselig Änderungen zwischen den verschiedenen Versionen abgeglichen werden müssen um einen Verlust zu verhindern. Bei den von uns genutzten Produkten entfiel diese Arbeit gänzlich und konnte somit in produktive Arbeit umgesetzt werden.

Eine weitere wichtige Funktionalität, die uns besonders bei der Erstellung der Dokumentation eine große Hilfe gewesen ist, ist die Möglichkeit jegliche Elemente in Dokumenten, also sowohl Text als auch Grafiken, mit Kommentaren zu versehen. Diese können entweder auf

einen Fehler hinweisen, oder eine angeregte Diskussion zu einem Thema anstoßen. So konnten wir jegliche Kommunikation über ein Dokument direkt darin führen, ohne in eine andere Umgebung, wie zum Beispiel ein E-Mail-Programm, zu wechseln. Diese Art und Weise zu arbeiten erfordert jedoch viel Vertrauen und klare Grenzen, da andere Mitarbeiter jederzeit Zugriff auf die Arbeit aller anderen haben und, möglicherweise ungewollte, Änderungen vornehmen könnten. Falls das zu einem Problem ausarten sollte, bot uns Google Docs die Möglichkeit mit einem Berechtigungssystem für jedes Dokument einzugreifen.

Bei der ersten Verwendung von Google Docs fiel es uns schwer mit der Struktur von Google Docs umzugehen, da statt der herkömmlichen Ordner, wie man sie von einem Computer gewohnt ist, auf so genannte Sammlungen gesetzt wird. Diese wirken anfangs umständlich, beweisen sich nach einiger Zeit jedoch als ein nützliches Werkzeug um Dokumente zu ordnen.

Eine Funktion die Google Docs nicht bietet, die wir besonders bei der Erstellung dieses Dokumentes vermisst haben, ist die Kompatibilität zu Vorlagen, die in einem anderen Programm als Google Docs erstellt wurden. So war es uns nicht möglich, den von unserer Schule zur Verfügung gestellten Dokumentenstil zu übernehmen. Stattdessen musste dieser von Hand nachgeahmt werden.

4.2 Google Groups

Von Seiten der für die Genehmigung der Diplomarbeiten Verantwortlichen wurde von uns verlangt einen Mailverteiler einzurichten, der bei Bedarf E-Mails an sämtliche Mitarbeiter des Teams verteilt. Dies kam unseren Plänen für die Handhabung der internen Kommunikation entgegen, da wir geplant hatten, wichtige Mitteilungen mittels eines Mailvertailers schnell innerhalb des Teams zu verbreiten.

Unsere Ansprüche an einen solchen Verteiler waren nicht allzu hoch, da er nur gelegentlich genutzt werden sollte. Erstaunlicherweise bieten nur wenige Anbieter einen solchen Dienst kostenlos an, was uns die Entscheidung erleichterte Google Groups für diesen Zweck zu nutzen. Dieses Produkt erlaubt es innerhalb weniger Minuten eine Liste zu erstellen, zu der beliebig viele Teilnehmer hinzugefügt werden können. Wenn nun E-Mails an die eigens für diese Liste erstellte Adresse gesendet werden, werden diese an sämtliche eingetragene Mitglieder verteilt. In unserem Fall war es auch außenstehenden Parteien erlaubt E-Mails an unseren Verteiler zu versenden, was aber bei Bedarf auch unterbunden werden kann.

Im Falle unserer Diplomarbeit war es notwendig die verteilten E-Mails für außenstehende unsichtbar zu machen, da sie teilweise interne Informationen enthalten haben. Es wäre aber auch möglich bei Bedarf die verteilten E-Mails in Form eines Forum-ähnlichen Archivs zu veröffentlichen, um einen Einblick in den Verlauf des Projektes zu ermöglichen.

Wir entschieden uns die Teilnehmer des Verteilers auf das engste Team, sprich Mitarbeiter und Hauptbetreuer, zu begrenzen, da viele der versendeten E-Mails nur für die genannten Personen relevant waren und wir damit hofften, unnötigen E-Mail-Verkehr für unsere Nebenbetreuer zu minimieren. Diese wurden im Gegenzug bei Bedarf eigens per E-Mail kontaktiert und um Rat gebeten. Wie bereits erwähnt war es Außenstehenden, wie zum Beispiel Sponsoren und Interessierten, trotzdem möglich E-Mails an unser gesamtes Team zu versenden.

4.3 Google Talk

Zusätzlich zu dem im vorhergehenden Kapitel genannten E-Mail-Verkehr innerhalb des Teams, führten wir den Großteil der Kommunikation

zwischen den Mitarbeitern über ein weiteres Produkt von Google, namens Google Talk. Dieses ermöglichte uns, unserem engen Finanzplan entgegenkommend, kostenlos so genannte Sofortnachrichten zu versenden. Es war also ein Chat den wir als Echtzeit-Kommunikationsmittel bei Zusammenarbeit über das Internet verwenden konnten, um den Mehraufwand von E-Mails und ähnlichen Kommunikationswegen zu vermeiden und schnell Probleme lösen zu können.

Dank der Verwendung von kompatiblen Smartphones mit entsprechendem Internetzugang konnte jeder diesen Dienst fast rund um die Uhr verwenden um Fragen mit anderen Mitarbeitern abzuklären. Somit wurde niemand bei seiner Arbeit aufgehalten, weil er auf eine Antwort per E-Mail warten musste.

4.4 Google Sites

Eine weitere Rahmenbedingung verlangte es von uns, Informationen über unsere Diplomarbeit von Anfang an der Öffentlichkeit in Form einer Website zu präsentieren. Darauf sollte für Außenstehende ersichtlich sein, in welcher Phase sich das Projekt im Moment befindet, und was überhaupt dessen Ziele sind.

Die Suche nach einem geeigneten Anbieter wurde wiederum stark von unserem Finanzplan beeinflusst, weshalb wir uns sehr schnell darauf einigten, Google Sites zu verwenden.

Google Sites bietet die Möglichkeit kostenlos eine stark vereinfachte Web-Präsenz zu erstellen. Nachdem wir uns mit der Umgebung vertraut gemacht haben, erstellten wir ein schlichtes Design, welches uns und unserer Arbeitsweise ähnelt. Dank der von Google Sites zur Verfügung gestellten Oberfläche mussten wir dazu keine Zeit in möglicherweise aufwendiges HTML, CSS und JavaScript investieren, mit dem Webseiten

normalerweise erstellt werden. Außerdem konnten wir uns hier ebenfalls die Anschaffungskosten für einen eigenen Server sparen.

4.5 Java

Die Entscheidung in welcher Programmiersprache dieses Projekt umgesetzt werden sollte, fiel uns aufgrund der Tatsache, dass wir im Unterricht und auch privat mit Java arbeiten, sehr leicht. Somit konnten wir sowohl unsere Erfahrungen, als auch unsere bisherigen Resultate aus früheren Projekten in dieser Diplomarbeit einsetzen. Eines der umfangreicheren Resultate, welches wir verwendet haben, ist die "common" genannte Bibliothek von Andreas Stefl. Sie enthält Klassen und Methoden, die immer wieder in Software-Projekten gebraucht werden, und hat uns somit einiges an Zeit erspart.

Die für Programmiersprachen längst nicht selbstverständliche, sehr voluminöse Standardbibliothek von Java hat uns ebenfalls etliche Stunden an Arbeit erspart. Sie umfasst zum Beispiel vorgefertigte Elemente für die Oberfläche von Programmen, aber auch Hilfsmittel um die Arbeit im Hintergrund eines Programmes zu erleichtern.

Ein weiterer wichtiger Faktor, der für Java spricht, ist Interoperabilität, also die Möglichkeit ein und den selben Code auf allen gängigen Plattformen, wie Linux, Windows und Mac OS auszuführen, und im Falle von Java das gesamte Programm in einer einzigen ausführbaren Datei auszuliefern, ohne dem Benutzer eine langwierige Installation aufzuzwingen. Somit ist es möglich unser Programm zum Beispiel auf einen USB-Stick zu kopieren und auf einem beliebigen Computer in ihrem Netzwerk auszuführen.

Eine Voraussetzung dafür Java-Programme und somit auch unser Programm ausführen zu können ist, dass Java, genauer gesagt das Java Runtime Environment, auf dem Zielcomputer installiert ist. Die meisten

Systeme werden aber mittlerweile von Haus aus damit ausgeliefert, beziehungsweise sind die nötigen Dateien für eine Installation kostenlos im Internet verfügbar.

Wie zuvor erwähnt ist ein Vorteil von Java ein Programm in einer einzigen Datei an den Benutzer auszuliefern. Das stimmt, birgt jedoch einige Fallen und Einschränkungen, wie wir im Laufe der Diplomarbeit feststellen mussten. So ist es möglich den selbst geschriebenen Code in eine einzige Datei zu exportieren, jedoch ist es nicht ohne weiterer Arbeit möglich, ein Programm samt Dritt-Bibliotheken zu exportieren. Da unser Programm leider in letztere Kategorie fällt, wie im Kapitel "4.9 Bibliotheken" genauer nachgelesen werden kann, mussten wir einen Weg finden, eben diese Bibliotheken in unser Programm zu integrieren, ohne deren Lizenzen zu verletzen.

Bei der Suche nach einer passenden Lösung sind wir auf die Grenzen von Java gestoßen, nämlich, dass standardmäßig kein Weg vorgesehen ist, Bibliotheken in ein Programm zu integrieren. Dank der unzähligen Möglichkeiten die Java Programmierern bietet, gibt es genau dafür bereits eine Lösung von anderen Entwicklern. Da deren Funktionsweise tiefergehende Java-Kenntnisse voraussetzt, wird sie hier nicht näher besprochen. Die gewählten Lösungsansätze sind das Eclipse-Plugin "fatjar", zu finden unter fjep.sourceforge.net, und die in Eclipse eingebaute Funktion "Export as runnable jar archive", wobei wir ersteres verwendet haben, da es gegenüber der in Eclipse eingebauten Funktion viele weitere für uns relevante Anwendungsfälle unterstützt.

4.6 Git und GitHub

Git ist ein so genanntes Versionskontrollsystem, dass hauptsächlich darauf ausgerichtet ist bei der Entwicklung von Programmen zu assistieren. Der Zweck eines solchen Versionskontrollsystems ist es die Arbeit an einem Programm mehreren Entwicklern gleichzeitig zu

ermöglichen. Nachdem diese ihre Änderungen an dem Programm vorgenommen haben, werden die veränderten Daten auf einen zentralen Server hochgeladen, der diese Daten daraufhin den anderen Entwicklern bereitstellt. Um Konflikte zwischen den Änderungen verschiedener Entwickler zu verhindern, und somit ein reibungsloses Arbeiten zu ermöglichen, müssen Entwickler jegliche Konflikte mit den am Server gespeicherten Änderungen selber lösen, bevor sie ihre eigenen hochladen können.

Abgesehen von dem Konfliktmanagement erlaubt es Git sämtlichen Mitarbeitern, und je nach Lizenzierung des Projektes auch außenstehenden Interessierten, Einsicht in den aktuellen Stand des Codes zu nehmen und gegebenenfalls weitere Änderungen vorzunehmen.

Die nicht-lineare Natur von Git erlaubt es Entwicklern ihre Version des Projektes von dem ursprünglichen Projekt abzuspalten und eigene Änderungen niemals in dieses zurückzugeben. Diese baumartige Struktur von Git-Projekten kann aber auch dazu verwendet werden, an mehreren neuen Funktionen für ein Programm gleichzeitig zu arbeiten. In einem Projekt wie netCrawler kann die Arbeit an der Oberfläche und den "Innereien" des Programmes zum Beispiel auf mehrere Mitarbeiter aufgeteilt werden um die Fertigstellung zu beschleunigen. Etwas weniger abstrakt erklärt, wäre die Arbeit an ein und dem selben Kapitel in einem Buch für zwei Schriftsteller nicht möglich. Lässt man diese jedoch an zwei verschiedenen Kapiteln des Buches arbeiten, ist eine parallele Durchführung möglich. Das Problem bei Software-Projekten ist aber, dass im Laufe der Zeit meistens Fehler, beziehungsweise in einem Buch Rechtschreibfehler, gefunden werden. Wenn diese Fehler nun von einem anderen Mitarbeiter als jenem behoben werden, der eigentlich für diesen Teil des Programmes zuständig ist, muss dafür gesorgt werden,

dass die nötigen Änderungen um den Fehler auszubessern auch wirklich aufgenommen werden, und nicht von dem zuständigen Mitarbeiter unabsichtlich überschrieben werden. Um dies zu bewerkstelligen zwingt Git Mitarbeiter vor dem Hochladen sämtliche relevanten ausstehenden Änderungen einzupflegen.

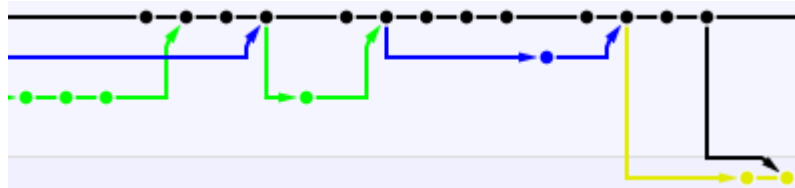


Abbildung 3: Ein Ausschnitt aus unserem Git-Graphen

Die zuvor erwähnte Baumstruktur entsteht durch zwei fundamentale Funktionen von Git, die es erlauben, sich von dem aktuellen Stand des Projektes abzuspalten. Da gäbe es die Möglichkeit entweder einen so genannten Fork, also eine Abspaltung in ein gänzlich neues Projekt, oder einen Branch, einen Zweig im bestehenden Projekt, zu erstellen. Außerdem wäre es möglich, dass alle Mitarbeiter in ein und dem selben Projekt und Branch arbeiten, was jedoch schnell zu ungewünschten Konflikten führen kann. Welche dieser Methoden letztendlich eingesetzt wird, hängt von den Vorlieben, Gewohnheiten und Kenntnissen der einzelnen Mitarbeiter ab, wobei Branches die sauberste, dafür aber auch aufwändigste Herangehensweise wäre.

In unserem Fall haben wir uns dazu entschieden für jeden Mitarbeiter einen eigenen Fork zu erstellen, da niemand von uns ein Experte im Umgang mit Git war. Rückblickend wäre eine Umsetzung mit Branches wahrscheinlich übersichtlicher gewesen.

Da Git, wie bereits erwähnt, einen zentralen Server voraussetzt, der sämtliche Änderungen bereithält, müsste im Normalfall, wie auch für E-Mail, Website und etliche weitere in diesem Kapitel besprochene Arbeitswerkzeuge, ein Server angeschafft werden, der diese Funktionen bereitstellt. Einige Firmen setzen zwar noch immer auf eigene Git-Server, für selbstständige Programmierer, Startups und kleinere Firmen ist aber ein GitHub, ein Anbieter für Git in der Cloud, mittlerweile beinahe ein Muss. Denn zusätzlich zu den eigentlichen Funktionen eines Git-Servers, werden einige wichtige Funktionen wie das Melden von Fehlern und weitere so genannte "soziale Funktionen" von GitHub angeboten. Besonders wichtig für uns war, dass dieser Dienst für Open-Source-Projekte, wie netCrawler, kostenlos ist.

Um mit dem Git-Server zu kommunizieren, kann je nach Bedarf, ein Tool verwendet werden, welches mittels eines Terminals bedient wird, oder ein Plugin für Eclipse, dass diese Aufgaben grafisch darstellt und somit vereinfachen soll. In unserem Team wurden beide Tools verwendet. Ersteres ist auf populären Plattformen einfach installierbar, letzteres nennt sich EGit und ist im Internet kostenlos erhältlich.

Um zu veranschaulichen wie Git zu verwenden ist, werden folgend die notwendigen Befehle besprochen. Angenommen das Projekt wurde bereits für Git eingerichtet und der Entwickler hat bereits Änderungen an dem Code vorgenommen, müssen diese in den Index aufgenommen werden. Dies geschieht am einfachsten mittels *git add -A*, wobei es für Fortgeschrittene in den meisten Fällen ratsam ist die geänderten Dateien einzeln mittels *git add [Datei]* hinzuzufügen. Anschließend wird der aktualisierte Index mit einer Beschreibung der getätigten Änderungen versehen, um später einen aussagekräftigen Versionsverlauf sämtlicher Änderungen zu erhalten. Änderungen samt Beschreibung nennt man einen Commit und werden mittels *git commit -m '[Beschreibung]'* erstellt. Zuallerletzt müssen noch die erstellten

Commits auf den Server hochgeladen werden, was mit *git push origin master* möglich ist.

Dieses Beispiel ist eine starke Vereinfachung, welche annimmt, dass ausschließlich ein Branch namens “master” verwendet wird, und keine unbekannten Änderungen am Server vermerkt sind. Außerdem setzt dieses Beispiel einiges an Vorkonfiguration voraus.

Alles in allem hat uns Git die Arbeit an diesem Projekt massiv erleichtert und uns somit nie die Entscheidung für Git über Alternativen wie SVN und CVS bereuen lassen. Besonders in den Phasen des Projektes, zu denen wir auf Hochtouren an dem Programm gearbeitet haben, hat uns die Arbeit mit Git sehr geholfen, und hin und wieder auch verblüfft.

4.7 Eclipse

Eclipse ist eine Entwicklungsumgebung, ursprünglich von IBM erfunden, die mittlerweile jeder Programmierer zumindest ausprobiert hat. Die gebotenen Funktionen reichen von den Anforderungen eines Anfängers, bis zu denen eines professionellen Programmierers. Wir haben sämtliche Entwicklung ebenfalls mithilfe von Eclipse durchgeführt, da wir alle bereits privat damit gute Erfahrungen gemacht haben. Aufgrund der bereits sehr lange andauernden Entwicklungszeit dieser Entwicklungsumgebung, kann mittlerweile beinahe jedes Detail daran nach Belieben geändert werden. Um bei der Arbeit mit Git, worüber mehr im Kapitel “4.6 Git und GitHub” zu lesen ist, keine unnötigen Konflikte aufgrund verschiedener Einstellungen hervorzurufen, bietet Eclipse die Möglichkeit, mithilfe eines so genannten Formatter, den Code an einen bestimmten Stil anzupassen. Unter anderem wird dabei die Einrückungstiefe im gesamten Code auf die Eingestellte angepasst. Für dieses Projekt haben wir einen eigenen Formatter konfiguriert, der den Programmierstil von uns allen in einem Konsens vereint.

4.8 Bibliotheken

4.8.1 gson

gson ist eine von Google in Form eines Open-Source-Projektes veröffentlichte Bibliothek, die die Interaktion mit JSON in Java ermöglicht. Mehr zu JSON kann in dem Kapitel “7.9 Datenablage” nachgelesen werden.

Diese Bibliothek ist eine der schnellsten und umfangreichsten Implementierungen, die es in diesem Gebiet gibt. Außerdem ist sie unter der “Apache License 2.0” veröffentlicht, was eine Verwendung in den verschiedensten Projekten ermöglicht.

4.8.2 MindTerm

MindTerm ist eine der am weitesten verbreiteten Bibliotheken für Kommunikation via SSH in Java. Mittlerweile kümmert sich ein Unternehmen um die Erhaltung dieses Projektes.

Diese Bibliothek wurde unter keiner Open-Source-Lizenz veröffentlicht. Stattdessen ist die Verwendung in nicht-kommerziellen, beziehungsweise beschränkt in kommerziellen Projekten kostenlos.

4.8.3 SNMP4J

Auch SNMP4J ist eine der wenigen Bibliotheken in ihrem Gebiet, die in diesem Umfang kostenlos erhältlich sind.

Bei diesem Projekt wird ebenfalls auf die “Apache License 2.0” gesetzt.

4.8.4 common

Bei der common-Bibliothek handelt es sich um das Werk von Andreas Stefl. Sie enthält häufig verwendete Code-Stücke um wiederkehrende Aufgaben zu erleichtern.

Eine Lizenz für dieses Projekt wurde noch nicht festgelegt, jedoch wurde es auf Absprache mit dessen Autor in unserer Diplomarbeit verwendet.

5 TESTWERKZEUGE - KER

Da wir uns erstmal auf Cisco Geräte konzentriert haben, gab es für uns nur wenige Möglichkeiten unsere Arbeit zu testen. Diese werden folgend erläutert.

5.1 Packet Tracer Telnet Bridge - STE

Packet Tracer ist ein Programm, welches von der Firma Cisco entwickelt wurde, um virtuelle Netzwerke aufbauen zu können ohne sich eine entsprechende Anzahl an Geräten selbst kaufen zu müssen. Es ist also primär als Lernprogramm gedacht und soll angehenden Netzwerkadministratoren dabei helfen, sich in der komplexen Welt des Netzwerkverkehrs zurechtzufinden.

Ganz wichtig an diesem Programm ist, dass es so gut wie alle Basisfunktionen der Geräte implementiert. Allerdings kann man von externen Geräten nicht auf diese Funktionen zugreifen. So auch nicht von einem System, auf dem netCrawler installiert ist.

5.1.1 Idee

Die grundsätzliche Idee zu dieser Schnittstelle kam uns durch die "Multiuser"-Funktion von Packet Tracer.

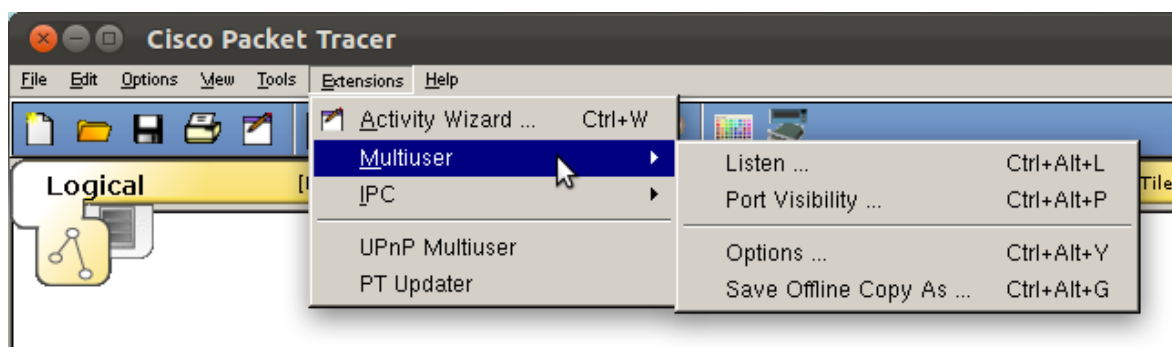


Abbildung 4: Die Multiuser-Funktion in Packet Tracer

Diese Funktion macht es möglich, zwei Instanzen des Programmes über ein reales Netzwerk zu verbinden. Da das Programm in irgendeiner Weise den virtuellen Netzwerkverkehr über das reale Netzwerk versenden muss, benötigt es ein Vermittlungsprotokoll. Dieses Protokoll kann jedoch genauso gut durch die Anwendung eines Dritten, wie unsere, verwendet werden.

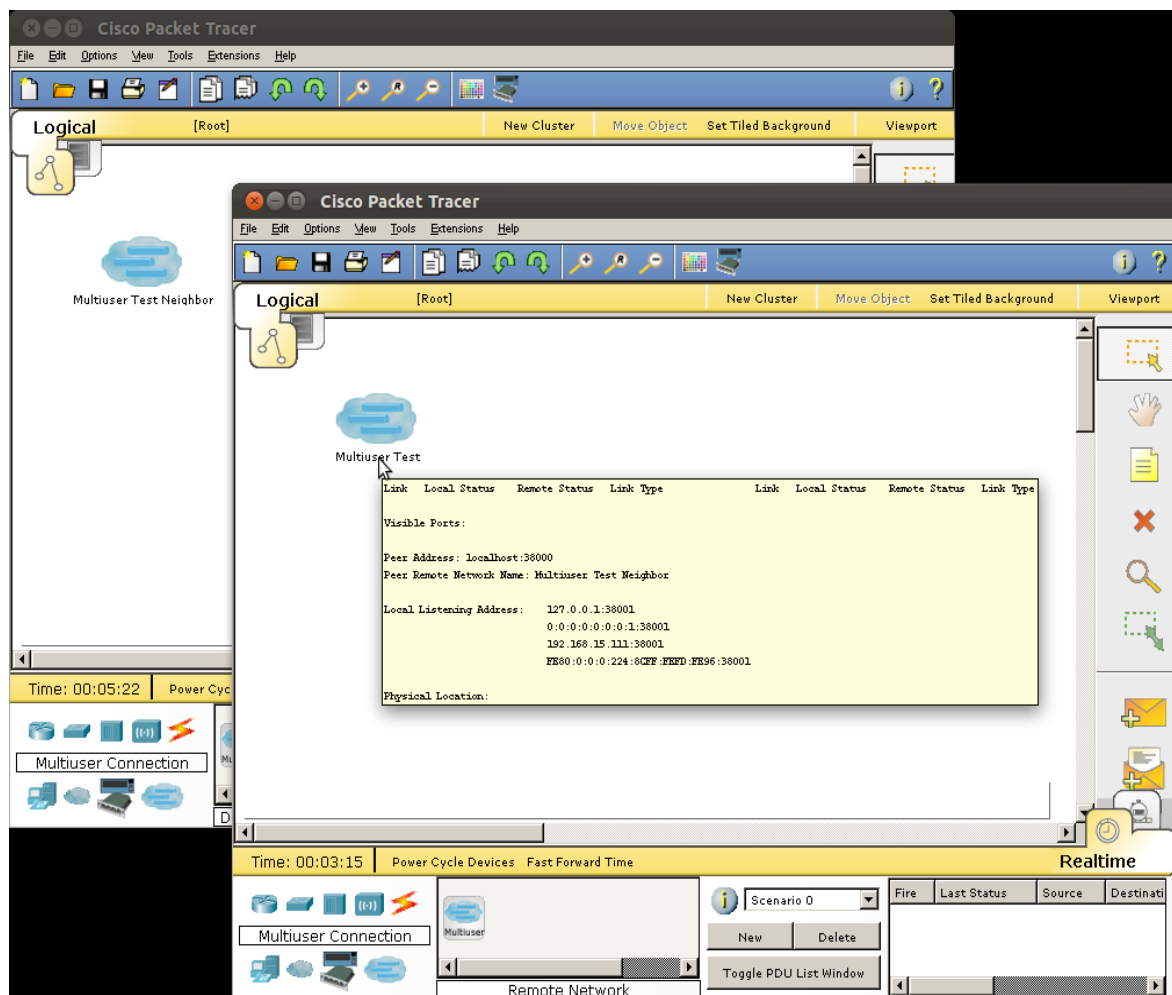


Abbildung 5: Multiuser-Connection in Packet Tracer erstellt und verbunden

Um unser Endprodukt netCrawler nun möglichst effektiv, und fern von allen "high-end" Netzwerkgeräten (beispielsweise zu Hause) testen zu können, hatten wir uns überlegt, ein Programm zu schreiben, das es

erlaubt Netzwerkverkehr in Packet Tracer einzuschleusen. Das ermöglicht uns Topologien in kürzester Zeit aufbauen, abspeichern und bei Bedarf wieder öffnen zu können.

5.1.2 Recherche

Cisco hat für die Kommunikation zwischen Packet Tracer Instanzen, das "Packet Tracer Message Protocol", PTMP definiert. Bei der Suche nach dem Vermittlungsprotokoll stießen wir auf PTMP. Dazu haben wir außerdem ein Dokument von Cisco gefunden.

PTMP operiert auf dem TCP/IP-Stack mit dem standardisierten Port 38000 und wurde für das Austauschen beliebiger Nachrichten zwischen zwei Packet Tracer Instanzen über das Netzwerk ausgelegt. PTMP stellt eine Übertragungsschicht dar, die jeder Nachricht einen Header zuteilt. Dieser besteht aus der Längenangabe und dem Typ des Pakets. Zusätzlich definiert PTMP den Verbindungsaufbau, der das Aushandeln der Verbindungseinstellungen (Kodierungs-, Kompressions-, Verschlüsselungs- und Authentifizierungsmethode) und die Authentifizierung beinhaltet.

Ein korrekter Verbindungsablauf sieht laut Definition wie folgt aus:

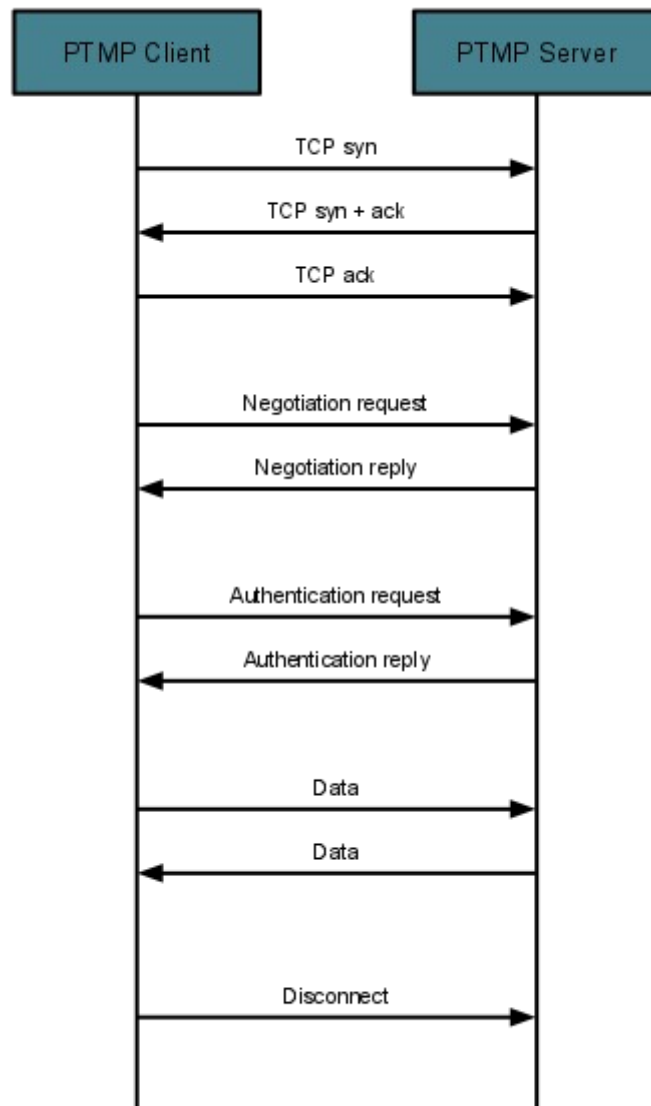


Abbildung 6: Ablauf einer PTMP-Verbindung

Wobei der Verbindungsabbau natürlich durch beide Seiten ausgelöst werden kann.

Dieses Protokoll alleine bietet uns jedoch noch keinen Zugriff auf das virtuelle Netzwerk. Dazu benötigen wir eine Multiuser-Verbindung, die direkt auf PTMP aufsetzt. Der Stack dieser Verbindung sieht so aus:

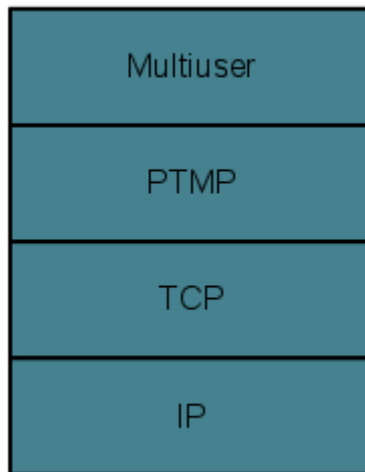


Abbildung 7: Der Stack der bei einer Multiuser-Verbindung durchlaufen wird

Das Multiuser-Protokoll besitzt leider keine Definition und man findet auch nichts dazu im Internet. Deshalb haben wir uns mit Ausgaben von Wireshark, ein Programm, das Netzwerkverkehr aufzeichnen kann, beschäftigt um einen eigenen Multiuser-Client zu programmieren, der so gut wie möglich dem des Packet Tracers gleicht.

Dieses Protokoll beinhaltet wiederum einen Verbindungsaufbau. Ist dieser vollzogen, werden die angeschlossenen "Links", also alle angeschlossenen Kabel an der "Cloud", von Packet Tracer bekanntgegeben. Jeder dieser "Links" hat eine eigene ID um sie beim Versenden von Netzwerkpaketen unterscheiden zu können. Der Verbindungsablauf sieht so aus:

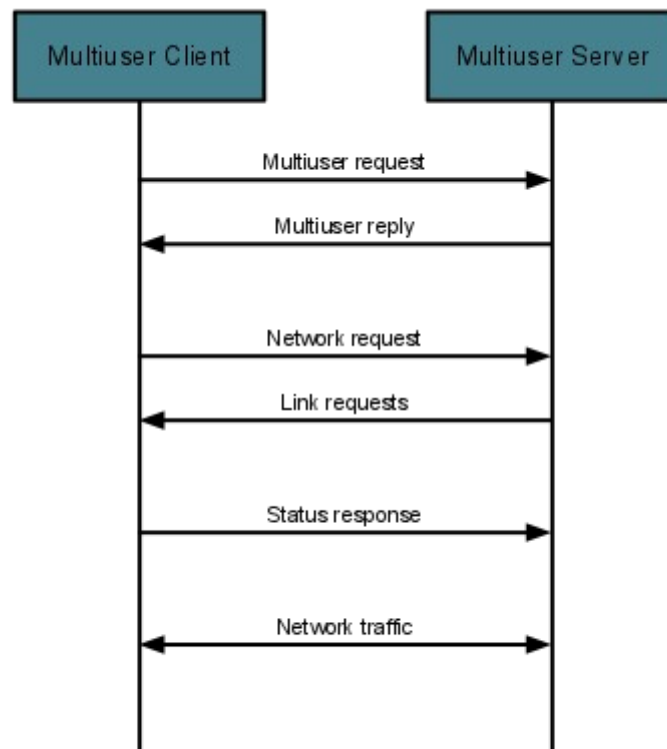


Abbildung 8: Ablauf einer Multiuser-Verbindung

Nachdem die Verbindung erfolgreich aufgebaut wurde, können die ersten Netzwerkpakete verschickt werden. Zu Anfang war dies ein guter Ausgangspunkt um die diversen Pakete aus Packet Tracer zu analysieren und zuzuordnen. Einige von ihnen wurden stark vereinfacht, bei anderen nur die Reihenfolge der Datenfelder geändert.

Hauptsächlich wurden ARP, IP und TCP analysiert, da wir nur diese für die Überbrückung von Telnet-Verbindungen benötigen. Letzteres Protokoll zeigte eine starke Unregelmäßigkeit gegenüber dem echten Standard auf. Da PTMP verschiedene Möglichkeiten der Kodierung besitzt, macht es wenig Sinn die übertragenen Daten nach Bytes zu zählen. Das betrifft speziell die Sequenznummern von TCP. Darauf wird aber im folgenden Abschnitt näher eingegangen.

Der Funktionsablauf einer Multiuser-Verbindung von Packet Tracer war damit geklärt. Das Modell dieser Verbindung sieht im Ganzen folgendermaßen aus:

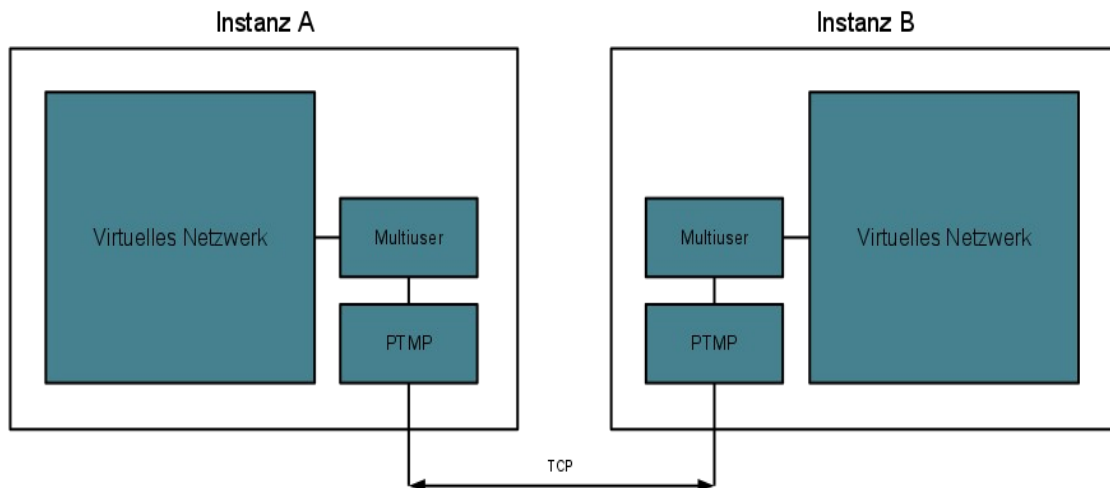


Abbildung 9: Überbrückung des virtuellen Netzwerkverkehrs von Packet Tracer

Durch das Auswechseln einer Packet Tracer Instanz durch unsere eigene Schnittstelle, ist es uns möglich beliebigen Netzwerkverkehr aus dem virtuellen Netzwerk zu empfangen, beziehungsweise in das virtuelle Netzwerk zu senden.

5.1.3 Implementierung

Bevor wir uns der Programmierung widmeten, haben wir uns ein grobes Konzept des Programmes überlegt. Zu aller erst benötigten wir eine Schnittstelle zu Packet Tracer. Diese formt unser Programm durch eine Multiuserverbindung.

Um Packet Tracer für netCrawler und Dritte bereitstellen zu können gibt es mehrere Möglichkeiten. Die beste wäre, den gesamten

Netzwerkverkehr mit dem von Packet Tracer zu überbrücken. Das hätte den großen Vorteil, dass netCrawler keinerlei Schnittstellen zu unserer Packet Tracer Bridge benötigen würde, da sie wie ein reales Netzwerk über eine gewöhnliche Netzwerkschnittstelle verfügbar ist.

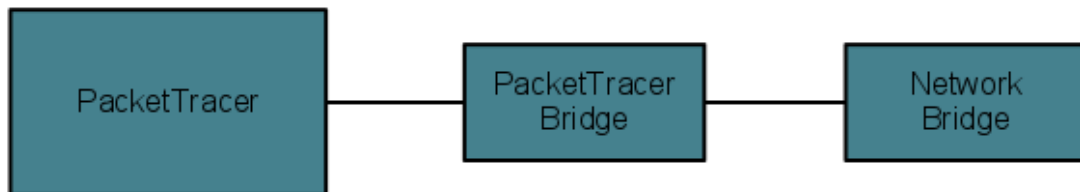


Abbildung 10: Packet Tracer Bridge als Schnittstelle zwischen Packet Tracer und realen Netzwerken

Die erste Hürde war es, Datenpakete der Netzwerkschicht zu generieren und zu versenden, beziehungsweise sie zu empfangen und zu dekodieren. Der Zugriff auf OSI-Schicht 2 ist jedoch nicht Anwendungsprogrammen, sondern viel mehr dem Kernel zugedacht. Dazu haben wir einen JNI Wrapper für die socket-Funktionen von Linux geschrieben. Diese Lösung der Netzwerküberbrückung ist zwar nicht optimal, aber sie funktioniert zumindest für Linux-Systeme. Wir versuchten außerdem die Entwickler von VirtualBox zu kontaktieren, da ihr Produkt eine Netzwerküberbrückung anbietet die auf allen Systemen funktioniert. Da jedoch eine Antwort ausblieb, mussten wir uns mit der eigenen Lösung begnügen.

Wie schon im vorherigen Abschnitt berichtet wurde, mussten wir leider feststellen, dass manche Datenpakete entschieden anders formatiert werden, als in einem realen Netzwerk. Obendrein überschneiden sie sich in ihrem Aufbau. Wenn diese Datenpakete also in das reale Netzwerk überbrückt werden sollen, müssen sie zuerst übersetzt werden. Ein

großes Problem dabei ist TCP. Dieses agiert im Bezug auf die Sequenznummern anders und hätte eine verbindungsorientierte Übersetzung gefordert. Dieses Vorhaben hätte jedoch jeden Zeitrahmen gesprengt und wir einigten uns auf eine einfachere, wenn auch nicht so dynamische Lösung.

Unser zweiter Lösungsansatz war es, einen virtuellen Computer an Packet Tracer anzuschließen, dem es möglich ist, eine Telnet-Verbindung aufzubauen. Da wir dabei das virtuelle Netzwerk nicht verlassen, mussten wir uns auch nur an die Regeln dessen halten. Die TCP-Sequenznummern waren also dabei kein Problem mehr. Alle benötigten Protokolle dafür waren ebenfalls schon analysiert, was die Implementierung sehr schnell voranschreiten ließ. Ein grobes Konzept dieser Lösung sieht folgendermaßen aus:

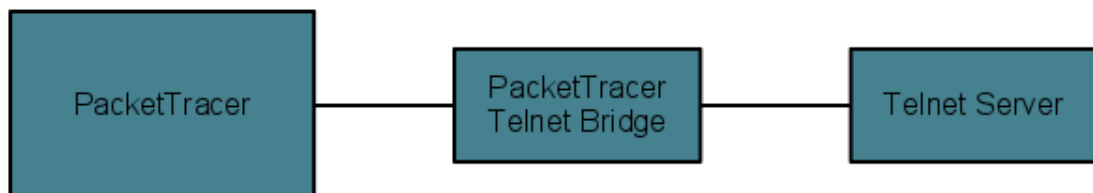


Abbildung 11: Packet Tracer Telnet Bridge als Schnittstelle zwischen Packet Tracer und einem Telnet Server

Wir benutzen einen virtuellen Computer, der eine Telnet-Verbindung aufbauen kann. Dieser hat wie gewöhnlich eine IP-Adresse und ein Standardgateway. Wenn nun eine Verbindung zum lokalen Telnet-Server aufgebaut wird, verbindet sich der virtuelle Computer mit einem vordefinierten Ziel in Packet Tracer. Dieser Zielcomputer kann zwar während der Ausführung der Telnet Bridge nicht gewechselt werden,

jedoch reicht uns diese Funktion zum Testen unseres Kommandozeilenparsers von netCrawler vollkommen aus.

5.2 Cisco Systems

Die Verwendung von Cisco Geräten bietet eine sehr umfangreiche und einfache Möglichkeit der Konfiguration. Außerdem ist unser Projekt schlussendlich auf diese Geräte spezialisiert.

Die Implementierung reicht von Telnet und SSH, SNMP, CDP und LLDP bis zu sämtlichen Routing Protokollen wie RIP, EIGRP, OSPF und BGP.

Das einzige Problem besteht darin, dass diese Testumgebung ausschließlich in der Schule verfügbar ist und somit die Entwicklung des Programms verzögern würde. Deswegen wurde diese Variante nur zum Testen von bestimmten Meilensteinen verwendet.

5.2.1 Konfiguration

```
ena

clock set 00:00:00 Jan 1 2000

conf t
hostname TestRouter
ip domain-name netcrawler.at

username cisco password cisco
username cisco privilege 15

crypto key generate rsa
yes
768

ip ssh version 2
```

```
snmp-server community netCrawler rw

line con 0
logging sync
login local
exit
line vty 0 4
login local
exit

int fa 0/0
ip address 192.168.0.254 255.255.255.0
no shutdown
exit

ip dhcp pool local
network 192.168.0.0 /24
default-router 192.168.0.254
exit

exit

write
```

6 TECHNOLOGIEN - KER

Um die festgelegten Ziele für unser Programm umsetzen zu können, benötigen wir Kenntnisse über die Funktionen, Schwachstellen und Abläufe verschiedenster Protokolle und Technologien. Dieses Kapitel beschäftigt sich mit dem von uns gesammelten Wissen und versucht es für Nachprojekte und außenstehende zugänglich zu machen.

6.1 SNMP

Allgemein

Das Simple Network Management Protocol ist ein Protokoll, welches es ermöglicht, Geräte in einem Netzwerk einfacher zu verwalten. Zurzeit gibt es drei Versionen des Protokolls, welche sich nur leicht voneinander unterscheiden. Die Unterschiede der Protokolle werden später näher beschrieben.

Geschichte

SNMP ist die Weiterentwicklung des 1987 entwickelten SGMP, das Simple Gateway Management Protocol, und wurde von der IETF, der Internet Engineering Task Force, entwickelt. SGMP konnte zwar Gateways überwachen, jedoch war dies die einzige Fähigkeit des Protokolls. Dies führte dazu, dass ein komplexeres Protokoll, mit mehreren Einsatzmöglichkeiten, benötigt wurde.

SNMP ist darauf ausgelegt, einfach und schnell einsetzbar zu sein und es somit auch auf kleineren Netzwerken anwendbar zu machen. Wichtig ist dabei, dass es die Leistung des Netzes nicht beeinträchtigt.

Funktionsweise

SNMP besteht grundsätzlich aus fünf Elementen:

- Management Station
- Management Agents

- MIB (Management Information Base)
- Managed Objects
- ASN.1

Die Funktion des Protokolls ist leicht erklärt: Auf jedem Gerät, welches über SNMP verwaltet werden soll, läuft ein Management Agent. Dieser wartet auf Abfragen oder Befehle der Management Station. Eine Station hat eine eigene Datenbank, in welcher die Geräte aufgelistet sind, die von dieser Station aus verwaltet und überwacht werden. Des Weiteren kann eine SNMP Struktur ein oder mehrere SNMP-Proxies enthalten. Diese sind eine Mittelschicht zwischen Station und Agent. Die Station sieht einen Proxy als Agent, während ein Agent die Proxy als seine Station ansieht. So kann eine Station mehrere Proxies verwalten, welche wiederum mehrere Agents überwachen. Dies dient dazu, dass der Administrator einen besseren Überblick über die Geräte innerhalb des Netzes hat.

Jede Information, welche per SNMP abgefragt oder verändert werden kann, wird Managed Object genannt. Diese Objects sind in einer lokalen MIB, Management Information Base, auf jedem Gerät gespeichert. Die MIB ist standardisiert und somit weltweit eindeutig.

Sie ist jedoch keine wirkliche Datenbank, in der Daten gespeichert sind, man findet dort nur eine Beschreibung, wo die gewünschten Daten zu finden sind. Die wirklichen Daten sind auf dem System abgespeichert. Es gibt jedoch nicht nur die Standard-MIB, sondern auch herstellerepezifische MIBs. Diese sind ebenso eindeutig, jedoch kann man dadurch Informationen von zum Beispiel Cisco Geräten auslesen, welche man bei anderen Herstellern nicht auslesen kann.

Die MIB ist in einer Baumstruktur aufgebaut und mithilfe des OID, Object Identifier, kann jedes Object in jeder MIB über den gleichen Pfad gefunden werden.

Die Objektbeschreibungen der Informationen sind in ASN.1 notiert. ASN.1 bedeutet Abstract Syntax Notation One und ist eine standardisierte, systemunabhängige Syntax.

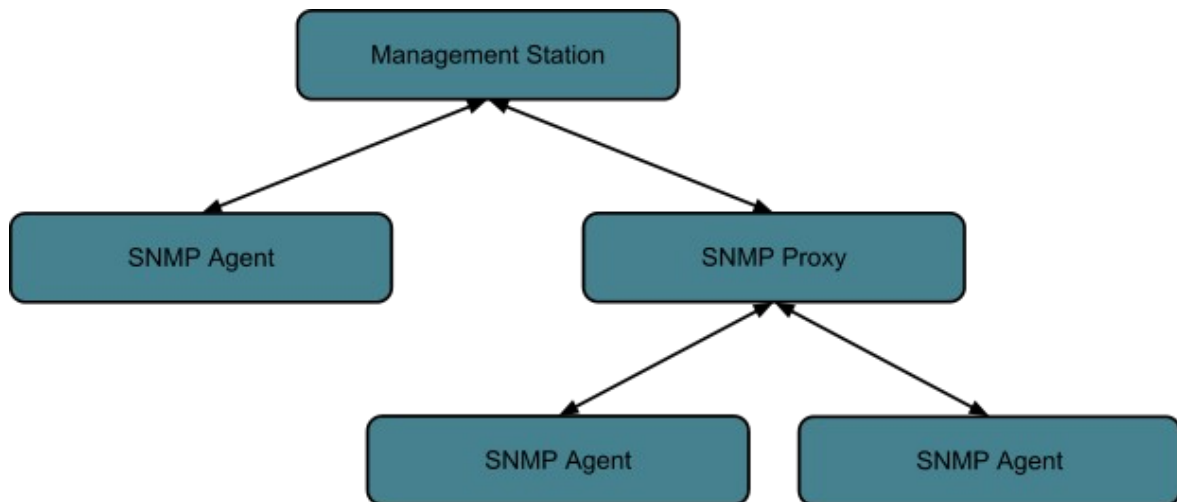


Abbildung 12: Mögliche Struktur eines Netzwerkes von SNMP-Geräten

6.1.1 SNMPv1

SNMPv1 basiert auf UDP, User Datagram Protocol, welches verbindungslos arbeitet. Jedoch ist diese erste Version von SNMP nicht sicher, da es noch keine Authentifizierungsmöglichkeit bietet. Daher wurde das Protokoll oft zur Überwachung, nicht aber zur Steuerung von Netzen verwendet. Ein weiteres Problem ist, dass die Station für jede Abfrage den Befehl noch einmal schicken muss. Dies erhöht den Traffic, vor allem bei einer großen Datenmenge.

Die erste Version des Protokolls kennt fünf Nachrichtentypen:

getRequest

Ist die einfache Abfrage eines Wertes aus der lokalen MIB des Agents.

Dieser Befehl wird von der Station an den Agent geschickt.

getResponse

Ist die Antwort auf ein getRequest mit den Werten und wird vom Agent an die Station geschickt.

getNextRequest

Dieser Befehl dient dazu, nicht den abgefragten Wert zu erhalten, sondern den in der MIB stehenden, direkten Nachfolger.

setRequest

Damit wird ein Wert in der MIB des Agents eingetragen oder geändert. Als Antwort sendet der Agent ein getResponse.

trap

Dies ist der einzige Befehl, welcher, unabhängig von der Station, vom Agent aus geschickt wird. Er dient lediglich dazu, bei Änderung der MIB eine Benachrichtigung an den Manager zu senden. Dieser Befehl ist unzuverlässig, da keinerlei Bestätigung zurückgesendet wird, ob der Befehl angekommen ist, oder nicht.

6.1.2 SNMPv2

Obwohl sich trotz der mangelnden Sicherheit in SNMPv1, das Protokoll schnell verbreitete, musste eine Lösung für die Fehler und Probleme gefunden werden. So wurde mit der Weiterentwicklung, also mit der Entwicklung von SNMPv2 begonnen.

Das Ziel von SNMPv2 war es die großen Sicherheitslücken zu schließen. Jedoch schlug dies fehl, da die Maßnahmen entweder zu kompliziert für die Implementierung waren, oder noch zu mangelhaft für die praktische Anwendung. Daher gibt es auch in SNMPv2 keine wirklichen Sicherheitsrichtlinien.

Gravierende Änderungen jedoch war die Einführung zwei neuer Nachrichtentypen:

getBulkRequest

Dieser Befehl ist die Weiterentwicklung von getRequest und dient dazu, größere Datenmengen effizienter zu beziehen.

informRequest

Dieser Befehl ist prinzipiell eine trap Nachricht, jedoch wird informRequest nicht vom Agent verschickt, sondern vom Manager an den Agent. Der Befehl kann zum Beispiel dazu dienen, dass der Agent den Manager über einen Fehler informiert. Außerdem sendet der Manager bei einem empfangenen informRequest ein Acknowledgement, eine Empfangs-Bestätigung, zurück an den Agent.

Eine weitere wichtige Änderung war die Möglichkeit, SNMPv2 auch auf OSI-Protokoll basierten Netzen zu verwenden. SNMPv1 kann nur auf IP-basierten Netzen verwendet werden.

6.1.3 SNMPv3

Da SNMPv3 so komplex ist, kann man nicht mehr von "simple" sprechen. Durch die vielen Funktionen dieser Version, ist vor allem die Performance beeinträchtigt. Außerdem wird durch diese Komplexität die Implementierung auf einfachen Geräten erschwert.

In SNMPv3 wurde jedoch erreicht, was in den vorigen Versionen des Protokolls nicht implementiert werden konnte: Sicherheit wurde durch folgende Mechanismen eingeführt:

Verschlüsselung

Der Inhalt der verschickten Pakete ist verschlüsselt und somit nur für den eigentlichen Empfänger lesbar.

SNMPv3 verschlüsselt mittels folgender Algorithmen:

- DES, der Data Encryption Standard, ist ein symmetrisches Verschlüsselungsverfahren
- AES, der Advanced Encryption Standard, ist ein symmetrisches Verschlüsselungsverfahren und gilt als Nachfolger von DES.

Authentifizierung

Es wird sichergestellt, dass die Nachrichten nur mit dem gewünschten Empfänger ausgetauscht werden.

Die Authentifizierung wird durch folgende Methoden umgesetzt:

- MD5, Message Digest Algorithm 5, ist eine Hashfunktion, welche aus einer Nachricht einen 128 Bit Hashwert erzeugt.
- SHA, Secure Hash Algorithm, erzeugt aus einer Nachricht einen 160 Bit Hashwert.

Integrität

Es wird sichergestellt, dass die Nachricht am Weg zum Empfänger nicht verändert wurde.

Die Integrität wird durch die gleichen Hash-Funktionen umgesetzt, wie die Authentifizierung.

6.1.4 Workflow

Um SNMP verwenden zu können, muss es zunächst installiert, konfiguriert und der Dienst gestartet werden. Je nachdem, welche Version von SNMP verwendet werden soll, muss die entsprechende installiert werden.

Mit den oben genannten Befehlen können nun Informationen ausgelesen werden. In der folgenden Grafik wird gezeigt, wie eine Abfrage im Hintergrund abläuft.

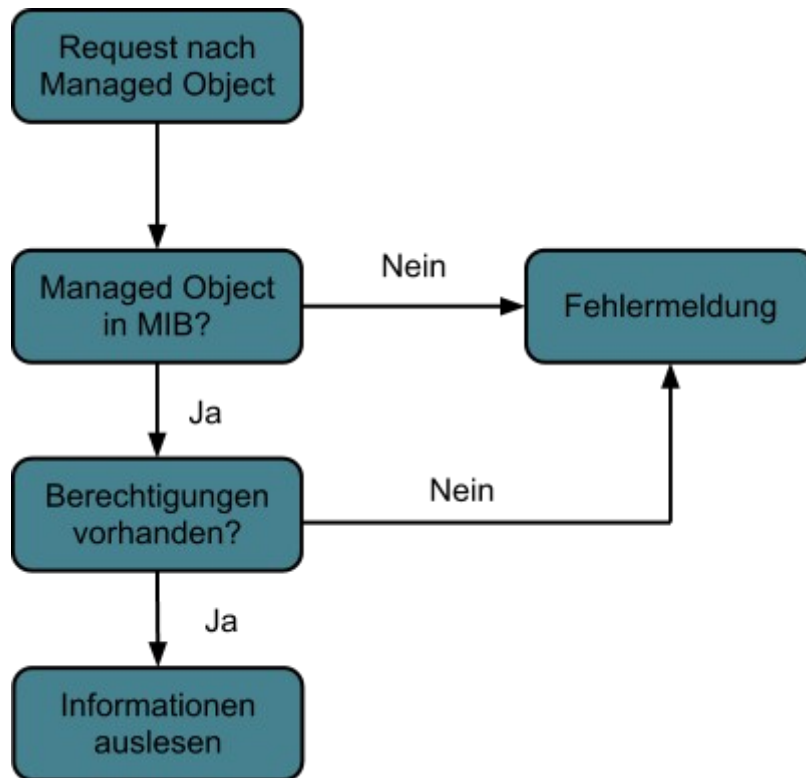


Abbildung 13: Ablauf einer Abfrage über SNMP

6.1.5 Problematik

Natürlich gibt es keine Vorteile ohne Nachteile, und bei einem komplexen Protokoll wie SNMP sind Probleme unausweichlich.

Ein großes Problem kann auch die MIB darstellen: sie ist zwar standardisiert, jedoch ist sie auch leicht durch Hersteller und Entwickler erweiterbar. Das ist eigentlich sehr nützlich, artet bei einem automatisierten Zugriff auf die MIB jedoch in mehr Arbeit aus. Ein Beispiel dafür ist der Hersteller Cisco, der teilweise für jedes Geräte-Modell oder auch jede einzelne Version der Firmware die MIB eigens erweitert. Grundsätzlich ist dies kein Problem sondern eher ein Vorteil, da man dadurch gerätespezifische Informationen auslesen kann. Jedoch ist damit sehr viel Recherche notwendig, um für jedes Gerät, beziehungsweise dessen Firmware die richtigen OIDs und die dazugehörigen Informationen herauszufinden. Bei jeglichen neuen Modellen oder Firmware-Updates muss erneut recherchiert werden und

jegliche Programme, die auf die MIB des Gerätes zugreifen, dementsprechend aktualisiert werden.

Dieses Problem besteht aber nicht nur zwischen verschiedenen Geräten eines Herstellers, sondern auch generell zwischen verschiedenen Herstellern. Angenommen Cisco würde die MIBs seiner Geräte jeweils um den selben Zweig erweitern, müsste trotzdem darauf geachtet werden den entsprechenden Zweig für andere Hersteller herauszufinden.

6.1.6 Praktisches Beispiel

Um SNMP besser verstehen zu können, wurde ein praktisches Beispiel durchgeführt. Dieses befasste sich damit, von einer Fedora-Maschine, mittels SNMP, Informationen von einer Ubuntu-Maschine auszulesen.

Grundlage

Für diese Struktur wurden zwei virtuelle Maschinen installiert. Beide wurden mittels einer Netzwerkbrücke mit dem Schulnetz verbunden. Somit waren diese Maschinen durch das Schulnetz erreichbar. Dieser Aufbau ist sehr geeignet, da man somit über SSH auf diese Maschinen zugreifen konnte, und man somit auch auf die Ausgaben des Terminals Zugriff hatte, was viel Zeit und Arbeit ersparte.

Fedora

Wie bereits erwähnt benötigte die virtuelle Maschine eine Verbindung mit dem Schulnetz. Nun musste nur noch die Interface-Konfiguration dahingehend geändert werden, dass die IP-Adresse automatisch bezogen wurde. Da das Programm VirtualBox das benötigte Interface erst hinzufügt, gibt es diese Konfigurationsdatei noch nicht. Daher muss diese erst erzeugt werden.

Dies wird unter Fedora folgendermaßen gemacht: die Datei `/etc/sysconfig/network-scripts/ifcfg-eth2` muss so aussehen:

```
DEVICE=eth2
```

```
BOOTPROTO=dhcp
```

```
ONBOOT=yes
```

Nachdem diese Konfiguration gesichert wurde, musste noch der network-Dienst neu gestartet werden:

```
service network restart
```

Mit dieser Konfiguration bezieht das Interface eth2 automatisch eine IP-Adresse vom Schul-DHCP-Server und ist somit auch mit dem Internet verbunden.

Um SNMP verwenden zu können, müssen zuerst folgende Pakete installiert werden:

- net-snmp
- net-snmpd-utils

Dies wird folgendermaßen gemacht:

```
sudo yum install net-snmp net-snmpd-utils
```

Nachdem diese Schritte abgeschlossen sind, befindet sich nun die Datei snmpd.conf im Ordner /etc/snmp/.

Bevor man diese Datei entsprechend ändert, ist es ratsam diese mit folgendem Befehl zu sichern:

```
cp snmpd.conf snmpd.conf.old
```

Falls bei den folgenden Schritten ein Fehler passiert, kann die Standard-Konfiguration einfach wiederhergestellt werden.

Wichtig ist zu Beginn, dass in der snmpd.conf Datei ein sogenannter Community-String eingefügt wird. Hierfür muss lediglich am Ende der Datei folgende Zeile eingefügt werden:

```
rocommunity public
```

Der Befehl `rocommunity` bedeutet "Read-Only Community" und `public` ist lediglich ein String, welcher häufig für diesen Zweck verwendet wird. Dieser String wird sozusagen als Authentifizierung verwendet. Wenn eine SNMP-Anfrage mit dem richtigen Community-String empfangen wird, werden die angeforderten Informationen gesendet. Ist der String falsch, wird die Anfrage verworfen.

Für eine einfache Abfrage ist die SNMP Installation und Konfiguration auf dem Fedora Rechner damit abgeschlossen. Nun kann man den SNMP-Dienst folgendermaßen starten:

```
service snmpd start
```

Um zu testen ob die Konfiguration funktioniert kann man eine SNMP-Abfrage von der eigenen MIB machen:

```
snmpget -v 2c -c public localhost 1.3.6.1.2.1.1.3.0
```

Mit diesem Befehl kann man zum Beispiel die Laufzeit eines Gerätes abfragen.

Als Ausgabe dieses Befehls erhält man:

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (414879)  
1:09:08.79
```

Aus dieser Ausgabe sieht man, dass das Gerät seit 1 Stunde und 9 Minuten läuft.

Ubuntu

Die SNMP-Konfiguration unter Ubuntu ist ähnlich der auf Fedora, erfordert jedoch weniger Konfigurationsarbeit.

Zu Beginn müssen `snmp` und `snmpd` installiert werden:

```
sudo apt-get install snmp snmpd
```

Wie bei Fedora wurde nun der Ordner `/etc/snmp` erzeugt, in welchem sich nun zwei Dateien befinden:

- snmp.conf
- snmpd.conf

Diese sollten ebenfalls zur Sicherheit kopiert werden.

Um nun Informationen auslesen zu können, muss nur eine Zeile geändert werden. In snmpd.conf befindet sich folgende Zeile:

```
agentAddress udp:127.0.0.1:161
```

Diese besagt, dass nur Anfragen vom localhost akzeptiert und bearbeitet werden. Jedoch wird benötigt, dass die Fedora-Maschine Zugriffsrechte hat. Also muss diese Zeile folgendermaßen geändert werden:

```
agentAddress udp:161
```

Nun werden nicht nur Abfragen vom localhost akzeptiert, sondern von allen Adressen.

Nun ist es möglich von der Fedora-Maschine, welche als Management Station fungiert, jegliche Informationen aus der MIB der Ubuntu-Maschine, dem Agent, abzufragen.

6.2 LLDP

6.2.1 Allgemein

Das Link Layer Discovery Protocol ist ein herstellerunabhängiges Protokoll, welches die Möglichkeit bietet adjazenten Geräten mitzuteilen, welche Nachbarn diese haben. Dadurch weiß jedes Gerät, wer dessen Nachbarn sind. Diese Updates werden nur in eine Richtung verschickt. Das bedeutet, dass jedes Gerät periodische Updates sendet, welche Informationen über das Gerät selbst enthalten. Jedes Gerät sendet und empfängt also Informationen gleichzeitig, ohne, dass eine Verbindung aufgebaut wird.

Diese Informationen werden, wenn bekanntgegeben, mittels des AgentX Protokolls, an einen SNMP Manager weitergegeben. Dadurch kann man mit SNMP auf die gesammelten Informationen zugreifen.

Das AgentX Protokoll ist eine modulare Erweiterung eines SNMP-Agents, und dient dazu Anfragen eines Master-Agents an einen Sub-Agent weiterzuleiten. Master- und Sub-Agents laufen auf einem Gerät. Der Master-Agent kommuniziert mit der Management Station und die Sub-Agents sind prinzipiell nur SNMP-Prozesse, welche die Informationen aus der MIB abfragen.

6.2.2 Workflow

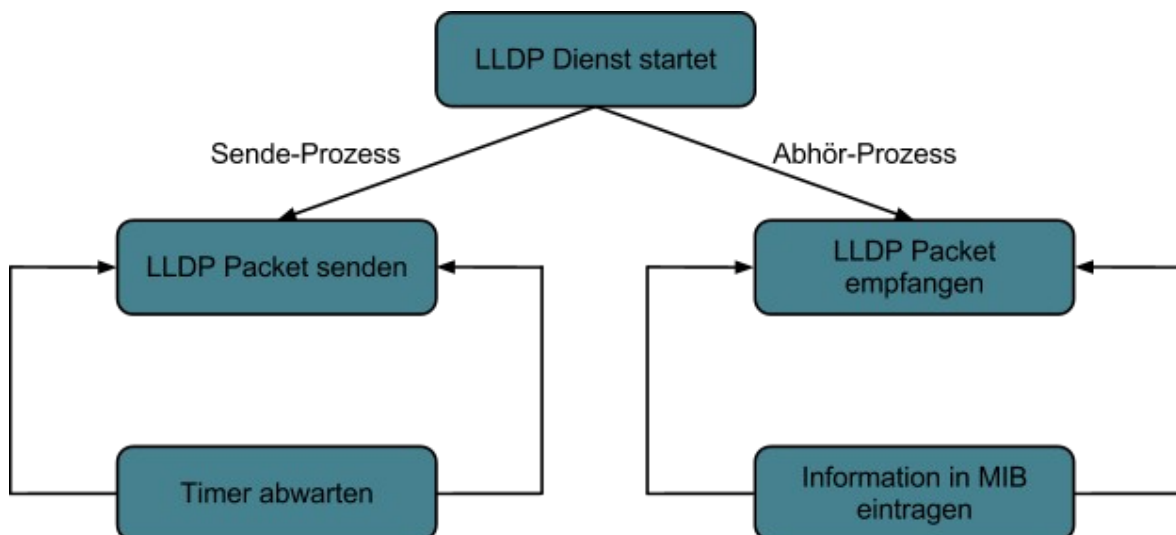


Abbildung 14: Ablauf eines LLDP-Dienstes

6.2.3 Problematik

Ein großes Problem bei LLDP liegt darin, dass dieses Protokoll nicht von allen Geräten verwendet werden kann. Außerdem muss die Implementierung AgentX beherrschen, damit mit dem SNMP-Service kommuniziert wird.

6.3 Telnet

6.3.1 Allgemein

Telecommunication Network ist ein verbindungsorientiertes Protokoll, welches über TCP operiert und dazu dient entfernte Computer über das Netzwerk zu steuern. Eine Telnet Verbindung besteht immer aus zwei Komponenten: Telnet-Client und Telnet-Server. Jedoch ist Telnet vollkommen unsicher und sollte durch SSH ersetzt werden. Trotz der Sicherheitslücken wird Telnet noch oft verwendet, zum Beispiel um die Verbindung zu einem Mail-Server zu testen. Dadurch kann man testen, ob eine Verbindung besteht und falls eine besteht, können mittels Telnet einfache Befehle ausgeführt werden um Mails zu empfangen beziehungsweise zu senden.

6.3.2 Funktionsweise

Wie bereits erwähnt gibt es bei Telnet immer einen Client und einen Server. Zu Beginn einer Telnet-Session wird der TCP-Handshake ausgeführt. Nach diesem Handshake ist die Verbindung aufgebaut. Es werden keine weiteren Mechanismen benötigt, um eine Sitzung aufzubauen.

6.3.3 Workflow

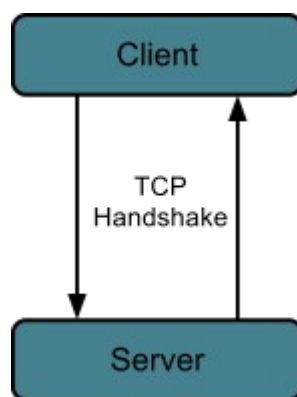


Abbildung 15:
Ablauf einer
Telnet-
Verbindung

6.3.4 Problematik

Da dieses Protokoll schon recht alt ist, gibt es keine Sicherheitsmaßnahmen. Dies zeigt sich vor allem dadurch, dass Telnet keine Verschlüsselung verwendet. Da mit einer hergestellten Verbindung ein Vollzugriff erlaubt wird, kann ein Angriff sehr leicht erfolgen und großen Schaden anrichten.

6.3.5 Praktisches Beispiel

Verbindung zu Mail-Server

Mit folgendem Befehl kann man eine Verbindung zu einem POP-Server, auch Mail Delivery Agent, testen. Dieser Server führt den E-Mail Empfang durch.

```
telnet pop3.web.de 110
```

Wenn der Server funktioniert und eine Verbindung aufgebaut werden kann, erhält man in etwa folgende Ausgabe:

```
Trying 212.227.17.177...
Connected to pop.web.net.
Escape character is '^]'.
+OK POP server ready H miweb109
```

Um andererseits einen SMTP-Server zu testen wird folgender Befehl verwendet:

```
telnet smtp.web.de 25
```

Ist die Verbindung erfolgreich wird folgendes ausgegeben:

```
Trying 217.72.192.157...
Connected to mail.web.net.
Escape character is '^]'.
220 web.de (mrweb002) Nemesis ESMTP Service ready
```

Mails empfangen

Waren die Tests erfolgreich, kann man nun seine E-Mails mit POP3, dem Post Office Protocol v3, abrufen.

Sobald man verbunden ist, wird man aufgefordert die Mail Adresse und anschließend das Passwort anzugeben, von der die Mails abgefragt werden sollen:

```
user testUser@gmx.at
+OK password required for user "testUser@web.at"
pass *****
```

Sobald man sich korrekt eingeloggt hat, erfährt man, wie viele Mails sich in der Inbox befinden.

```
+OK mailbox "testUser@web.at" has 182 messages (10867100
octets) H migmx102
```

Mit folgendem Befehl kann man sich eine bestimmte E-Mail mit der gewünschten Nummer anzeigen lassen, wobei die neueste Nachricht die höchste Nummer hat:

```
retr [Mail-Nummer]
```

Wenn man sich wieder ausloggen will, genügt folgende Eingabe:

```
quit
```

Mails versenden

Um sich mit dem Postausgangs-Server zu verbinden wird SMTP, Simple Mail Transport Protocol verwendet. Sobald man mit dem Server Verbunden ist, muss man diesen "begrüßen", was mit folgendem Befehl erledigt wird:

```
helo mail.web.net
```

Nun kann man bereits eine E-Mail schreiben. Zuerst muss angegeben werden, von welcher E-Mail die Nachricht verschickt wird. Danach wird der Empfänger eingetragen.

```
mail from: testUser@web.at  
rcpt to: testuser2@web.at
```

Nun wird angegeben, welche Mail Adresse in der Nachricht, als Sender und Empfänger stehen sollen.

```
From: User1Name <testUser@web.at>  
To: User2Name <testUser2@web.at>
```

Danach wird ein Betreff benötigt und erst danach kann man den eigentlichen Inhalt der E-Mail schreiben.

```
Subject: Betreff  
Content  
Content  
quit
```

Durch die Eingabe von quit wird die Nachricht gesendet und die Verbindung beendet.

6.4 SSH

6.4.1 Allgemein

Secure Shell wird, wie auch Telnet, dazu verwendet, eine Verbindung auf ein entferntes Gerät herzustellen. Im Gegensatz zu Telnet ist diese Verbindung jedoch verschlüsselt. Die Kommandozeilen-Outputs des entfernten Gerätes werden auf der Kommandozeile des eigenen Gerätes ausgegeben. So kann man dieses Gerät konfigurieren, als säße man "davor".

Es gibt inzwischen zwei Versionen von SSH. SSHv2 ist grundsätzlich ähnlich der ersten Version, jedoch unterstützt es 3DES, Triple-Data Encryption Standard, bei dem DES mehrmals durchgeführt wird, und AES, Advanced Encryption Standard, als weitere Verschlüsselungsalgorithmen. Standardmäßig wird AES, mit einer 128 Bit Schlüssellänge verwendet.

Der Vorteil von SSH gegenüber Telnet ist die Authentifizierung, welche mittels eines Public- und einem Private-Key ermöglicht wird.

6.4.2 SSHv2 Workflow

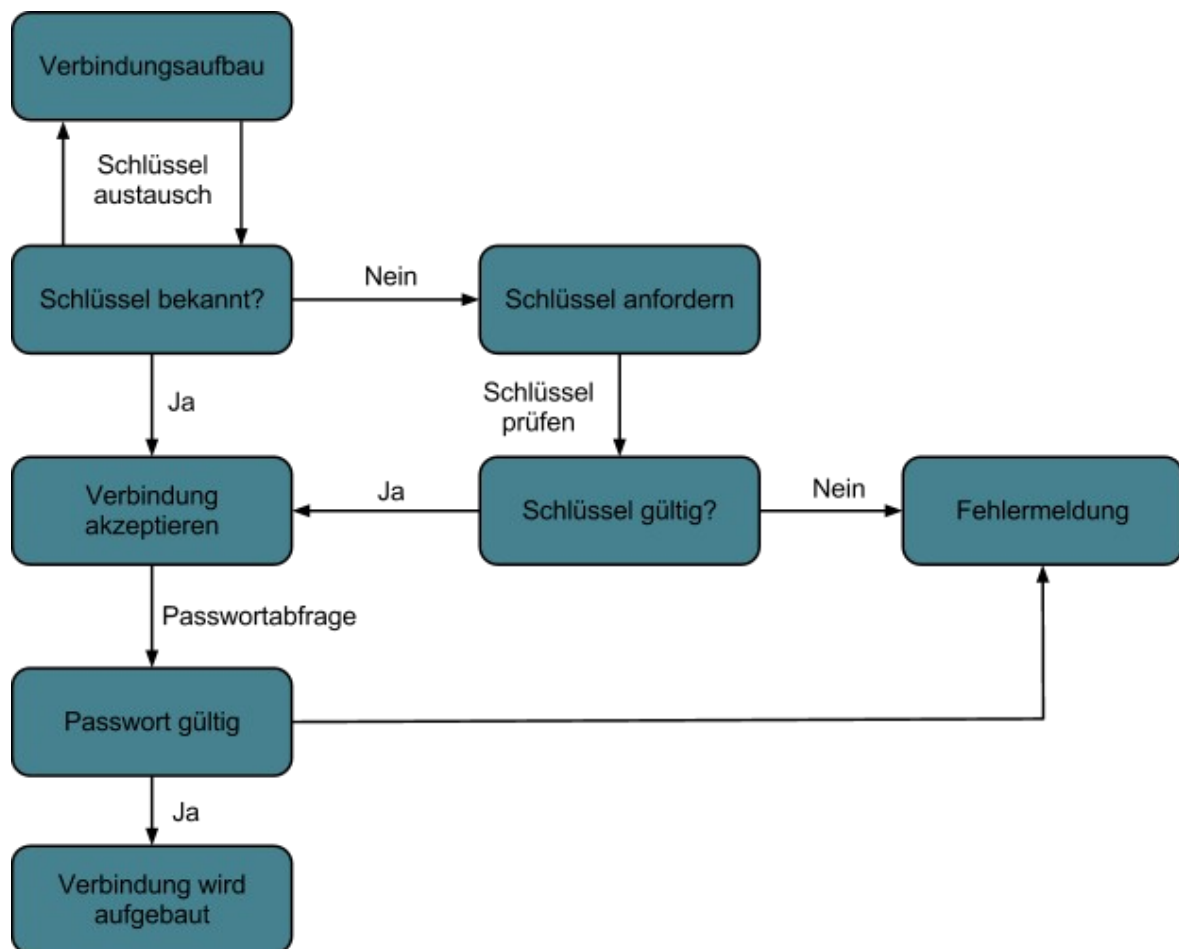


Abbildung 16: Ablauf einer SSH-Verbindung

6.4.3 Problematik

Bei SSHv1 gibt es das Problem, dass die Integritätsprüfung einige Schwachstellen aufweist und somit ist es möglich sich in eine SSH-Session "einzumischen", die übertragenen Daten also zu lesen und zu ändern, bevor sie den eigentlichen Empfänger erreichen. Dieses Problem besteht bei SSHv2 durch Verschlüsselung nicht mehr.

6.4.4 Praktisches Beispiel

Automatische User Anmeldung

Dieses Beispiel beschreibt, wie man SSH konfigurieren muss, um sich ohne Passwortabfrage mit einem anderen Rechner zu verbinden. Dies ist vor allem für SCP, Secure Copy, von Vorteil, da diese Konfiguration es ebenso ermöglicht, Dateien zu kopieren, sowie Skripte auszuführen. Dies wäre sehr langwierig, müsste man bei jeder Datei das Passwort eingeben.

Benutzt wurde eine Fedora-Maschine und eine Ubuntu-Maschine, von denen beide ein gebridgetes Netzwerk-Interface hatten, welches mit dem Schulnetz verbunden war.

Als Beispiel wird hier gezeigt, wie man sich von einer Ubuntu-Maschine mittels SSH auf eine Fedora-Maschine verbinden kann, ohne ein Passwort eingeben zu müssen.

Ubuntu

Um SSH mit diesem Rechner verwenden zu können musste zuvor openssh installiert werden:

```
sudo apt-get install openssh
```

Nun mussten, für den RSA-Schlüsselaustausch, der public- und der private-key erzeugt werden:

```
ssh-keygen
```

Nachdem der Befehl ausgeführt wurde, kann man den Speicherort für die Schlüsseldateien angeben, sowie das Passwort.

Nun kann mit dem Befehl

```
ssh-copy-id [Benutzername]@[IP-Adresse]
```

die eigene SSH-ID auf den anderen Rechner, in unserem Fall die Fedora-Maschine, kopiert werden. Somit wird der Ubuntu-Rechner mittels seiner ID gespeichert. Nach diesem Befehl wird der RSA-Fingerprint und das Passwort des Benutzers gespeichert. RSA steht für Rivest, Shamir, Adleman, welche die Erfinder des Verfahrens gelten. Durch die Sicherung des Fingerprints wird der Benutzer gekennzeichnet. Somit wird bei jeder Verbindung festgestellt, dass er auch der richtige Benutzer ist.

Will man sich das nächste Mal mittels SSH verbinden, wird in der Datei `~/.ssh/authorized_keys` gesucht, ob der Key des Rechners, welcher dem RSA-Fingerprint entspricht, gespeichert ist. Ist der Key gespeichert wird die Verbindung ohne Passwortabfrage, zugelassen. Ist der Key unbekannt wird ein Passwort benötigt.

6.5 CDP

6.5.1 Allgemein

Das Cisco Discovery Protocol wurde von Cisco entwickelt. Die aktuelle Version von CDP ist CDPv2, welches eine bessere Fehlererkennung ermöglicht. Da dieses Protokoll proprietär ist, wird es fast ausschließlich von Cisco-Geräten verwendet. Es dient dazu, ähnlich wie LLDP, die adjazenten Nachbarn eines Gerätes zu erkennen. Dadurch kennt jedes Gerät seine Nachbarn. Diese Informationen können aus der MIB ausgelesen werden. Es werden nur die unmittelbaren Nachbarn erkannt und in einer Tabelle gespeichert. Auf diese Tabelle kann unter anderem via SNMP zugegriffen werden.

Jedes Gerät sendet periodische Updates um die Datenbanken immer auf dem neuesten Stand zu halten. Bleibt jedoch ein Update aus und ist der Timer abgelaufen, werden die Informationen dieses Gerätes verworfen.

6.5.2 Workflow

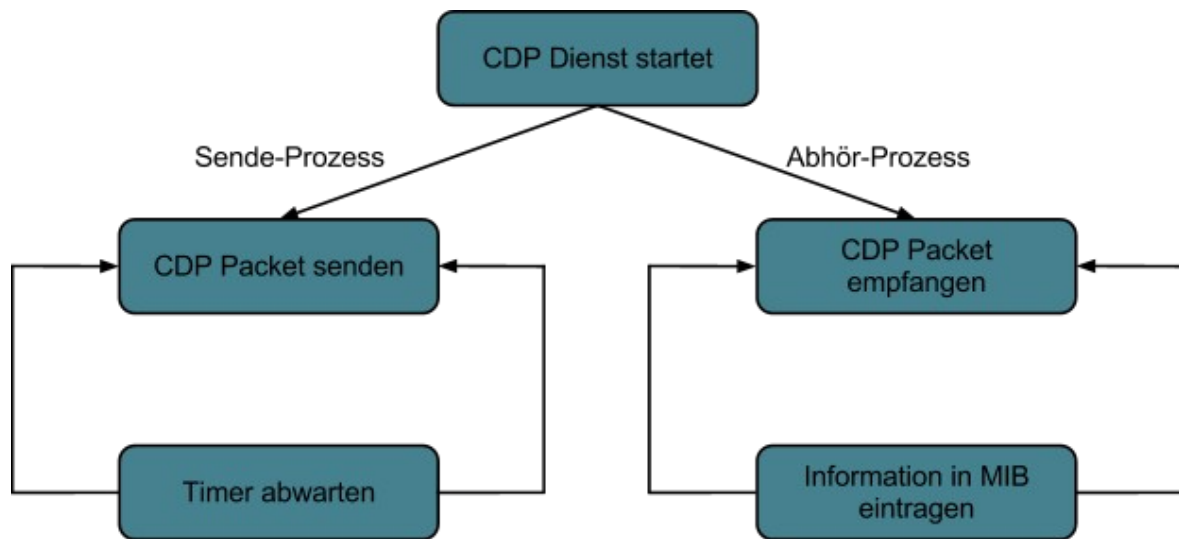


Abbildung 17: Ablauf eines CDP-Dienstes

6.5.3 Problematik

Jedoch kann CDP auch zu Problemen führen. Da CDP, wie gesagt, von Cisco proprietär entwickelt wurde, unterstützen nur Cisco-Geräte dieses Protokoll. CDP ist dahingehend unsicher, dass nur unverschlüsselte Pakete verschickt werden.

7 UMSETZUNG - STE

7.1 Voraussetzungen

Ein Programm gliedert sich meistens in mehrere Module. Viele dieser Module, die oft Anwendung finden, sind bereits durch Bibliotheken im Internet verfügbar. Zu Beginn der Planung eines Programms, muss man sich überlegen ob es bereits passende Bibliotheken gibt, oder ob man sich selbst der Implementierung widmen muss. Dabei spielen viele Faktoren eine Rolle. Für uns waren dabei die wichtigsten Kriterien: Lizenz und Aufbau beziehungsweise Handhabung der Bibliothek. Beispielsweise wäre es nicht sinnvoll gewesen SSH selbst zu implementieren, da der Aufwand nicht abschätzbar war. Außerdem wollten wir uns nicht eingehend mit SSH beschäftigen, sondern viel mehr mit dem Scannen von Netzwerken. Deshalb haben wir uns nach einer Bibliothek umgesehen, die unseren Wünschen entspricht. Sie sollte unter einer Open Source-Lizenz verfügbar sein, beide Versionen des Protokolls unterstützen und einen objektorientierten Aufbau haben. Dabei stießen wir auf MindTerm, welches unseren Erwartungen stand halten konnte. Das gleiche galt für SNMP. Dabei sind wir auf die Bibliothek SNMP4J gestoßen, welche die Versionen 1, 2c und 3 optimal umsetzt.

Eine weitere Hürde für unser Programm war die Serialisierung von Objekten. Wir stellten uns die Frage, wie wir am besten bestehende Topologien abspeichern könnten. Der erste Ansatz bestand darin die Serialisierung von Java zu verwenden. Diese ist jedoch binär kodiert und kann mit Hilfe eines Texteditors nicht ohne weiteres gelesen oder editiert werden. Eine weitere Idee war die Verwendung von XML. Dieses Format erschien uns jedoch als zu komplex. Schlussendlich stießen wir auf JSON, welches durch die Bibliothek gson optimal in Java eingebunden werden kann.

Das letzte Problem war das Darstellen eines Netzwerkgraphen, sowohl programmintern als auch in der grafischen Oberfläche. Da dieses Thema unsere Begeisterung weckte, widmeten wir uns selbst der Implementierung. Die Dokumentation für die programminterne Verwaltung des Graphen ist im Kapitel “7.7 Informationsverwaltung” zu finden. Jenes Modul, das den Graphen visualisiert, ist unter “7.8 Darstellung des Netzwerks” dokumentiert.

7.2 Programmablauf

In diesem Unterkapitel wollen wir kurz beschreiben welchen Ablauf unser Programm hat. Zu Beginn eines Suchlaufs muss bekannt gegeben werden, welches Gerät als Ursprung für die Suche verwendet werden soll. Anschließend wird eine geeignete Verbindung auf das Gerät aufgebaut. Je nachdem welche vom Benutzer bevorzugt wird und welche gerade verfügbar ist. Das Gerät wird dann durch beispielsweise die Seriennummer des Prozessors identifiziert. Als nächstes werden die Nachbargeräte ausgelesen, die beispielsweise durch CDP erfasst wurden. Zuletzt werden die restlichen Informationen, die im Kapitel “7.5 Wichtigste Geräteinformationen” zu finden sind, ausgelesen. Somit wäre ein Gerät fertig ausgelesen.

Anschließend wird eines der Nachbargeräte ausgewählt und wiederum eine geeignete Verbindung aufgebaut. Das Gerät wird wie zuvor identifiziert, die Nachbarinformationen werden analysiert und alle weiteren Informationen ausgelesen. Dieser Vorgang wiederholt sich so lange, bis das gesamte Netzwerk abgebildet ist.

Der Netzwerkgraph wird anschließend im Hauptfenster dargestellt und kann vom Benutzer angepasst werden.

Eine genauere Beschreibung des Vorgangs ist unter “7.7 Suchlauf” zu finden.

7.3 Erkennung der Geräte

Die Nachbarerkennung kann nicht durch netCrawler selbst durchgeführt werden. Daher müssen alle Netzwerkgeräte diese selbst unterstützen. Unser Programm fordert beim Auslesen der Informationen eines Geräts auch die Nachbarinformationen an, und analysiert diese. Dabei werden bis zum jetzigen Zeitpunkt CDP und LLDP unterstützt.

7.3.1 CDP

CDP wird auf Cisco Plattformen dazu verwendet um Nachbargeräte zu erkennen. Diese Informationen werden von einem Gerät durch einen Befehl, der auf der Kommandozeile abgesetzt wurde, oder eine SNMP-Abfrage ausgelesen.

Eine nähere Beschreibung zu diesem Protokoll ist unter "6.5 CDP" im Kapitel "6. Technologien" zu finden.

7.3.2 LLDP

Da LLDP ein standardisiertes Protokoll ist, wird es von vielen Netzwerkgeräten unterstützt. Die Nachbarinformationen werden direkt an die MIB von SNMP weitergegeben und können so abgefragt werden.

Eine nähere Beschreibung zu diesem Protokoll ist unter "6.2 LLDP" im Kapitel "6. Technologien" zu finden.

7.4 Auslesen der Geräteinformationen

Um die Informationen eines Geräts auslesen zu können benötigt man eine Verbindung. Grundsätzlich unterstützt netCrawler Kommandozeilen, die per Telnet, SSH, sowie SNMP erreicht werden können. Da sich unsere Testumgebung jedoch auf Geräte von Cisco beschränkt, konnten wir bisher nur einen Parser für Kommandozeilen

dieses Herstellers implementieren. Der Aufbau des Parsers ist jedoch sehr modular und kann jederzeit erweitert werden.

SNMP bietet hingegen einen transparenten Zugriff auf jedes Netzwerkgerät, das dieses Protokoll implementiert.

Es gibt also drei Verbindungswege zwischen denen netCrawler unterscheidet, aber im Endeffekt alle Daten konsistent ausliest. Die verschiedenen Verbindungsarten machen es möglich Geräte verschiedenster Hersteller und Funktion auszulesen.

7.4.1 Telnet

Telnet ist weit verbreitet und wird für die Verwaltung von Netzwerkgeräten eingesetzt. Dabei wird eine Kommandozeile bereitgestellt, die es möglich macht Befehle abzusetzen und deren Ausgabe auszulesen.

Wir mussten uns nun überlegen wie wir an die Informationen der Kommandozeile herankommen. Da diese eigentlich für die Bedienung von Menschen geschaffen ist, war dies nicht ohne weiteres innerhalb eines Programmes möglich. Das Problem besteht darin die Ausgabe von Befehlen zu filtern, da sie beispielsweise durch die Prompt verfälscht wird.

Um dieses Problem zu lösen, haben wir einen Parser implementiert, der diese Aufgabe übernimmt. Er synchronisiert Ausgabe und Eingabe und verhält sich im Endeffekt wie ein Benutzer, der die Kommandozeile bedient.

Dies warf jedoch ein weiteres Problem auf. Die Kommandozeilen unterscheiden sich von Hersteller zu Hersteller. Deshalb gestalteten wir den Aufbau des Parsers sehr modular um schnelle Erweiterungen möglich zu machen. Außerdem muss der Parser unterscheiden auf welcher Kommandozeile er sich gerade befindet. Beim Aufbau der

Verbindung ist nämlich nicht klar, was für ein System sich dahinter befindet. Aufgrund von Zeitmangel haben wir uns darauf geeinigt nur Geräte von Cisco auf Basis der Kommandozeile zu unterstützen.

Das letzte Problem, das es zu lösen gab, war die Handhabung der Authentifizierung. Da Telnet nicht die Möglichkeit bietet dies auf Protokollebene zu verwalten, muss der Parser selbst die Verbindung authentifizieren. Aus Zeitgründen wurde dafür eine schlichte Mustererkennung eingebaut, die auf Schlüsselwörter wie "Username" und "Password" reagiert.

7.4.2 SSH

Im Gegensatz zu Telnet wird von SSH die Authentifizierung der Verbindung übernommen. Bis auf diese kleine Besonderheit kann der Parser, den wir bereits bei Telnet eingesetzt hatten, auch bei SSH verwendet werden.

Die Einschränkungen bestehen jedoch weiterhin. Es werden bis zum jetzigen Zeitpunkt nur Geräte von Cisco unterstützt.

7.4.3 SNMP

SNMP bietet eine vollkommen transparente Verbindung auf ein Gerät, im Bezug auf Hersteller und System. Das macht es möglich, die Informationen sehr elegant auszulesen und ohne großen Aufwand in das Datenmodell einzutragen.

7.5 Wichtigste Geräteinformationen

Eine der wichtigsten Aufgaben von netCrawler ist die Beschaffung von Informationen. Daher haben wir uns schon zu Anfangs unseres Projekts überlegt, welche Informationen wichtig sind und wie sie aus den Geräten ausgelesen werden können.

Ganz zu Beginn mussten wir Informationen suchen, die auf jedem Gerät eindeutig sind. Diese sollten in weiterer Folge dazu dienen das Gerät zu identifizieren. Da es möglich ist, dass ein Gerät im Zuge eines Suchlaufs öfter ausgelesen wird, mussten wir dem entgegenwirken. Ein wichtiger Punkt dabei ist auch, dass die Entstehung von Schleifen verhindert wird.

Bei Geräten von Cisco gibt es beispielsweise die Processor board ID. Eine Information die auf jedem Gerät unterschiedlich ist. Diese wird von uns verwendet, um Cisco Router zu identifizieren. Bei Switches dieses Herstellers gibt es hingegen die System serial number. Beide Kennungen sind in der Ausgabe des show version Befehls zu finden.

SNMP bietet dazu einen eigenen Eintrag in der MIB, dessen Wert auf jedem Gerät unterschiedlich ist. Dieser nennt sich snmpEngineID und ist über die OID 1.3.6.1.6.3.10.2.1.1 erreichbar.

Der nächste große Punkt war das Finden von Nachbarinformationen. Unter Systemen von Cisco kann man alle Informationen die per CDP erfasst wurden mit dem Befehl show cdp neighbors anzeigen. Per SNMP sind diese unter der OID 1.3.6.1.4.1.9.9.23.1.2.1 zu finden.

Die Nachbarinformationen von LLDP können unter Cisco mit dem Befehl show lldp ausgelesen werden. Per SNMP sind diese Informationen unter der OID 1.3.6.1.2.1.1.1 zu finden.

Es gibt jedoch auch noch Informationen, die für den Benutzer wichtig sind. Beispielsweise der Name des Geräts, oder wie lange dieses schon in Betrieb ist, oder die IP-Adressen, unter denen das Gerät erreichbar ist und viele mehr. Dazu haben wir uns eine Tabelle erstellt, die alle wichtigen Informationen beinhaltet.

Information	OID	Cisco Befehl
Hostname	1.3.6.1.2.1.1.5	show running-config
Systembezeichnung	1.3.6.1.2.1.1.1	show version
Gerätetyp	1.3.6.1.2.1.1.7	show version
Uptime	1.3.6.1.2.1.1.3	show version
Featureset	1.3.6.1.2.1.1.7	show version
Netzwerkschnittstellen	1.3.6.1.2.1.2.2.1	show ip interface
Management IP-Adressen	1.3.6.1.2.1.4.20.1	show ip interface brief

Tabelle 1: Die wichtigsten Informationen eines Gerätes

7.6 Suchlauf

Der Suchlauf ist die wichtigste Funktion von netCrawler. Sie macht es dem Administrator möglich, mit einer Hand voll Informationen das gesamte Netzwerk auszulesen. Es gibt viele Methoden ein Netzwerk nach Geräten zu durchsuchen. Beispielsweise könnte man alle möglichen IP-Adressen mit ICMP nach Antwort abfragen. Da Methoden dieser Art jedoch sehr langsam sind, mussten wir uns etwas anderes überlegen.

Dabei fiel uns ein, dass es spezielle Protokolle für die Nachbarerkennung von Geräten gibt. Unter Systemen von Cisco übernimmt dies CDP. Es gibt aber auch ein standardisiertes Protokoll, LLDP. Diese Protokolle agieren auf jedem Gerät im Netzwerk, das dieses unterstützt. Dabei werden die Informationen der Nachbargeräte auf jedem Gerät gespeichert und können mit diversen Protokollen und Befehlen abgefragt werden.

Um ein Netzwerk zu durchsuchen, benötigen wir zuerst die IP-Adresse zu einem Netzwerkgerät. Davon ausgehend, lesen wir alle Nachbarn

dieses Geräts aus. Weiters werden dann die Nachbarn der Nachbarn ausgelesen und so fort.

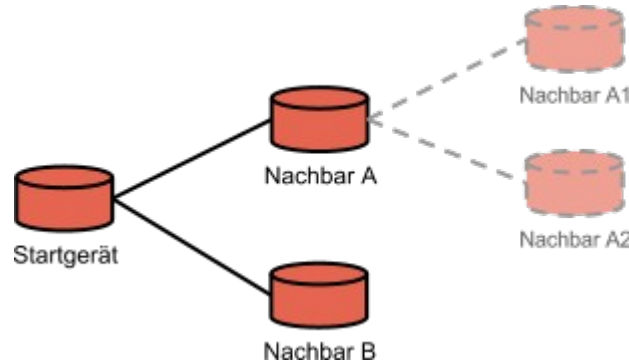


Abbildung 18: Der Suchlauf visualisiert

Damit war der prinzipielle Ablauf des Suchvorgangs geklärt. Dabei kam jedoch ein Problem zum Vorschein. Zwei Nachbargeräte haben sich jeweils gegenseitig in ihrer Nachbartabelle eingetragen. Wird also immer eine Verbindung auf eines der Geräte in der Nachbartabelle aufgebaut, so kommt es zu einer Schleife. Dieses Problem könnte prinzipiell mit Split Horizon gelöst werden. Jedoch würde das Problem der Schleifenentwicklung in Netzwerken mit Kreisen trotzdem auftreten. Deshalb haben wir uns etwas anderes überlegt. Jedes Gerät wird zu Anfang der Verbindung identifiziert. Welche Informationen dazu verwendet werden kann im Kapitel "7.4 Wichtigste Geräteinformationen" nachgelesen werden. Ist dieses Gerät schon einmal im Zuge des Suchlaufs ausgelesen worden, bricht das Programm den Vorgang des Auslesens noch vor der Nachbarerkennung ab. Damit ist gewährleistet, dass sich keine Schleifen bilden können.

7.6.1 Optimierung

Da ein Suchlauf von großen Netzwerken viel Zeit in Anspruch nehmen könnte, oder durch die Verwendung von SSH stark verlangsamt wird, haben wir uns überlegt wie wir diesen schneller gestalten könnten. Das

Schlüsselwort dafür war Multithreading. Jeder Auslesevorgang eines Geräts kann in einer eigenen Verbindung laufen. Das beschleunigt den ganzen Suchlauf enorm und die Implementierung war nicht weiter aufwändig.

7.7 Informationsverwaltung

Eine organisierte prozessinterne Informationsverwaltung ist für die Effizienz eines Programms sehr wichtig. Daher haben wir uns früh mit der Frage befasst, wie wir später eine gesamte Netzwerktopologie abspeichern können.

Dazu mussten wir ein Netzwerk in seine logischen Bestandteile zerlegen: die Netzwerkgeräte, deren Netzwerkschnittstellen und ihren Verbindungen, den Netzworkkabeln. Jeder Bestandteil hat eine Vielzahl von verschiedenen Eigenschaften. Ein Netzwerkgerät hat beispielsweise oft einen Namen oder mehrere IP-Adressen, unter denen es erreichbar ist. Diese Bestandteile können jedoch auch hersteller- und produktabhängig sein. Ein Netzwerkgerät kann beispielsweise ein Switch oder ein Router sein. Diese unterscheiden sich jedoch grundsätzlich und besitzen andere Informationen.

Um die Informationsverwaltung möglichst einfach zu halten, haben wir die Informationen der einzelnen Bestandteile von jenen der endgültigen Struktur des Netzwerks, also der Netzwerktopologie, getrennt.

7.7.1 Netzwerkbestandteile

Wie schon zuvor erwähnt, besteht ein Netzwerk aus verschiedenen Bestandteilen. Jedes von ihnen kann mehrere Abwandlungen haben. Ein Netzwerkgerät kann beispielsweise ein Router oder ein Switch sein. Eine Netzwerkschnittstelle kann eine Ethernet-Schnittstelle oder eine serielle Schnittstelle sein. Für jede dieser Schnittstellen sind Netzwerkmedien standardisiert.

In einem Objektmodell löst man solche Problemstellungen für gewöhnlich durch Vererbung. Verschiedene Hersteller von Geräten bieten unterschiedliche Informationen, die wir möglichst modular abspeichern und auslesen wollten. Würde man also das Konzept der Vererbung anwenden, würde beispielsweise die Klasse Router von der Klasse Netzwerkgerät erben. Wenn es sich in der Realität jedoch um einen Cisco-Router handelt, der spezielle Informationen bereitstellt, die wir unbedingt benötigen, können wir diese nicht unter Router ablegen, da andere Geräte von anderen Herstellern nicht über diese Daten verfügen. Dazu müssten wir eine neue Klasse, CiscoRouter einführen, die von der Klasse Router erbt. Alle Cisco-Geräte besitzen jedoch ebenfalls untereinander gemeinsame Informationen. Die IOS-Version beispielsweise ist auf jedem Gerät zu finden. In unserem Beispiel müsste CiscoRouter von Router und CiscoGerät erben. Das ist in Java jedoch nicht möglich, da Polymorphismus nicht implementiert ist. Das veranlasste uns dazu, uns ein neues Konzept zu überlegen.

Unsere Lösung war ein schlichtes assoziatives Datenfeld, welches erweitert werden kann und als Schlüssel eine Zeichenkette verwendet. Dazu erschufen wir die Basisklasse NetworkModel. Von ihr erbten in weiterer Folge die Klassen NetworkDevice, NetworkInterface und NetworkCable.

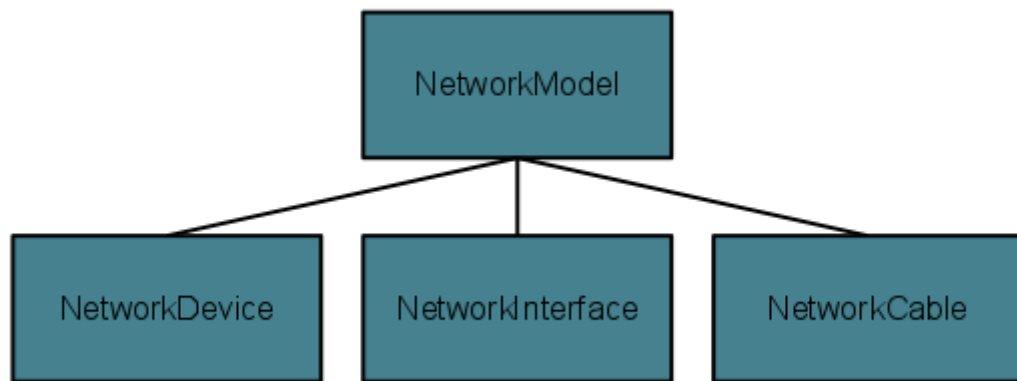


Abbildung 19: Visualisierung der verwendeten Hierarchie bei den Basisklassen

Die Klasse `NetworkDevice` übergibt jedem ihrer Objekte bei der Initialisierung Basisattribute, die im zuvor erwähnten assoziativen Datenfeld abgelegt werden. Ein Netzwerkgerät besitzt, wie bereits erwähnt, einen Namen. Da dieser nichts weiter als eine Zeichenfolge ist, wird er in Java in einem Objekt der Klasse `String` abgelegt. Dieses Attribut ist mit dem Schlüssel `device.name` belegt und kann damit abgefragt oder geändert werden.

Jede dieser Klassen hat außerdem eine Basisklasse für Erweiterungen. Als Wurzel dient dazu die Klasse `NetworkModelExtension`.

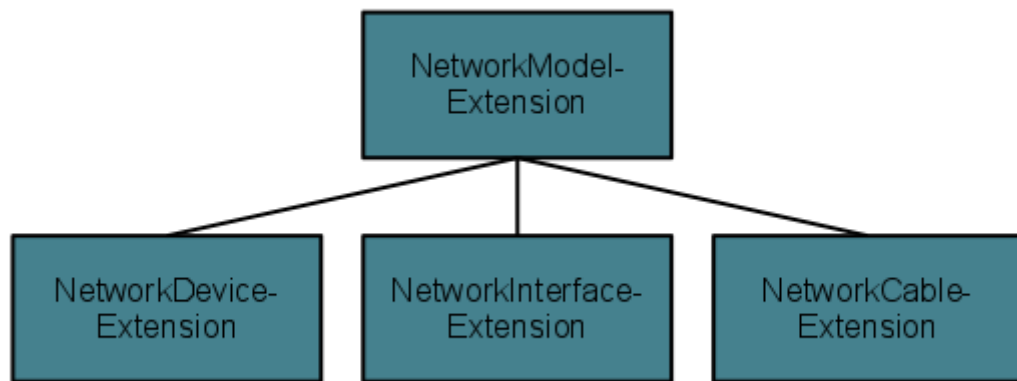


Abbildung 20: Visualisierung der verwendeten Hierarchie bei den Erweiterungsklassen

Eine Erweiterung besteht wiederum aus einem assoziativen Datenfeld, das in weiterer Folge dem zu erweiternden Objekt übergeben wird.

Aufgrund der gegebenen Abhängigkeiten begannen wir mit der Modellierung der Basisklassen, NetworkModel und NetworkModelExtension.

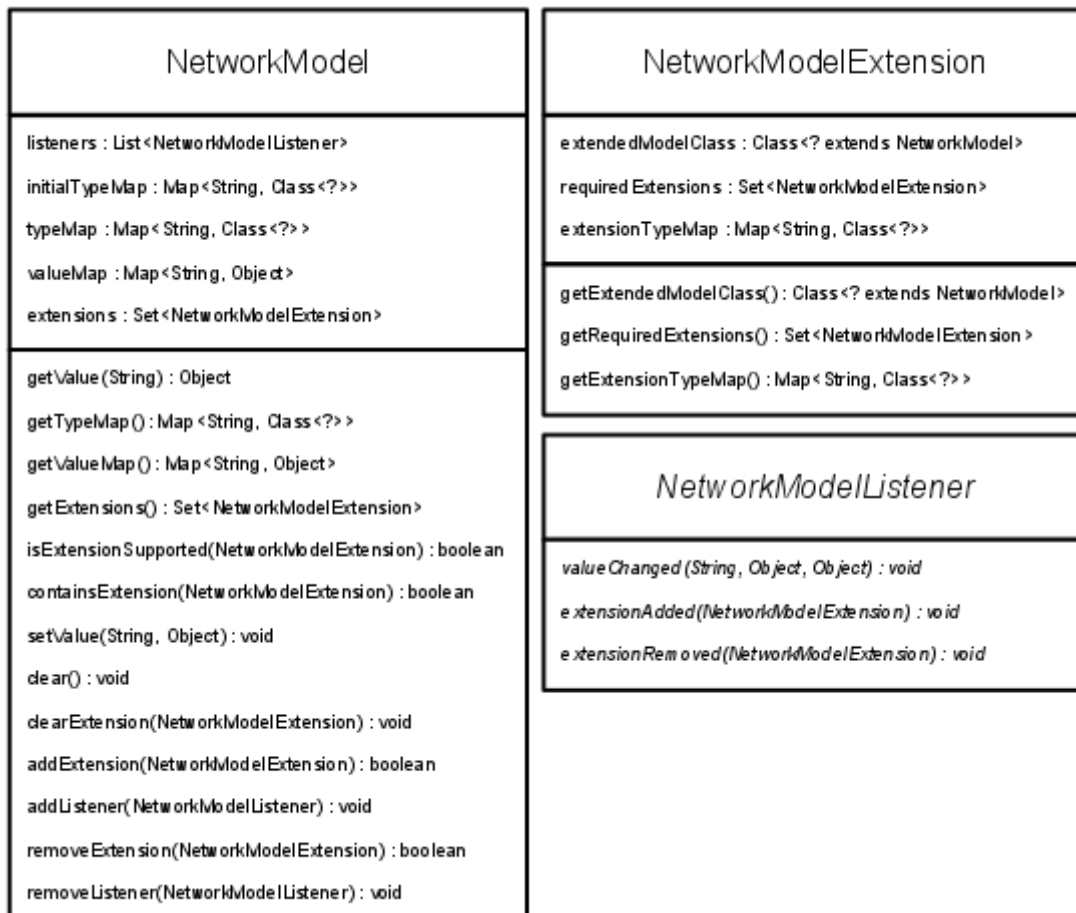


Abbildung 21: Die Klassen NetworkModel, NetworkModelExtension und NetworkModelListener in einer Übersicht

Es gibt also die Basisklasse NetworkModel. Ihre wichtigsten Attribute sind typeMap, valueMap und extensions. Ersteres enthält alle gültigen Schlüssel mit ihrem zugehörigen Datentyp. Es dient der Validierung eines hinzugefügten Objekts. Das Attribut valueMap speichert alle Informationen im Zusammenhang mit ihrem Schlüssel. Und zuletzt das Attribut extension, dieses enthält alle hinzugefügten Erweiterungen.

Um nun eine Ethernet-Erweiterung für die Basisklasse NetworkInterface zu schreiben, damit die MAC-Adresse der Netzwerkschnittstelle abgelegt werden kann, reichen folgende Zeilen:

```
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

import at.andiwand.library.network.mac.MACAddress;
import at.netcrawler.network.model.NetworkInterfaceExtension;

public class EthernetInterfaceExtension extends
NetworkInterfaceExtension {

    public static final Map<String, Class<?>>
EXTENSION_TYPE_MAP;

    public static final String ADDRESS =
"interface.ethernet.address";

    public static final Class<MACAddress> ADDRESS_TYPE =
MACAddress.class;

    static {

        Map<String, Class<?>> map = new HashMap<String,
Class<?>>();

        map.put(ADDRESS, ADDRESS_TYPE);

        EXTENSION_TYPE_MAP =
Collections.unmodifiableMap(map);
    }

    public EthernetInterfaceExtension() {
        super(EXTENSION_TYPE_MAP);
    }
}
```

7.7.2 Netzwerktopologie

Der nächste Schritt war die letztendliche Verwaltung der Struktur des Netzwerks.

Da sich ein Netzwerk optimal als Graph darstellen lässt, entschieden wir uns dazu, diese Gegebenheit als Ausgangspunkt zu verwenden. Die Ecken repräsentieren die Netzwerkgeräte und die Kanten deren

Verbindungen. Ein Problem stellten jedoch die Netzwerkschnittstellen der Geräte dar. Wie sollten diese in einem Graphen untergebracht werden, beziehungsweise sollten sie das überhaupt?

Unser erster Ansatz war es, die Netzwerkschnittstellen ebenfalls als Ecken zu betrachten. Das machte den Graphen an sich jedoch sehr unhandlich, da es zwei verschiedene Arten von Ecken und Kanten gab: jene Ecken, die ein Netzwerkgerät oder eine Netzwerkschnittstelle darstellen, und jene Kanten, die ein Netzwerkkabel repräsentieren oder die Zugehörigkeit einer Netzwerkschnittstelle zu dessen Gerät.

Der nächste Ansatz war, das Betrachten eine Netzwerkschnittstelle nicht direkt als Element eines Graphen, sondern mehr als eine weitere Eigenschaft eines Netzwerkgeräts.

Dabei stießen wir auf ein weiteres Problem, das wir zuvor übersehen hatten: es gibt Netzwerkkabel, die nicht nur zwei Enden besitzen. Auch wenn diese Technologien schon lange obsolet sind, nahmen wir uns diesem Problem an.

Nach einer ausgiebigen Suche im Internet stießen wir auf so genannte Hypergraphen. Sei bestehen aus Ecken und Hyperkanten. Die Besonderheit einer Hyperkante ist es, dass sie mehrere Ecken verbinden kann.

Alles in allem erschien uns diese Lösung als sehr elegant und wir begannen sie in ein objektorientiertes Konzept umzusetzen.

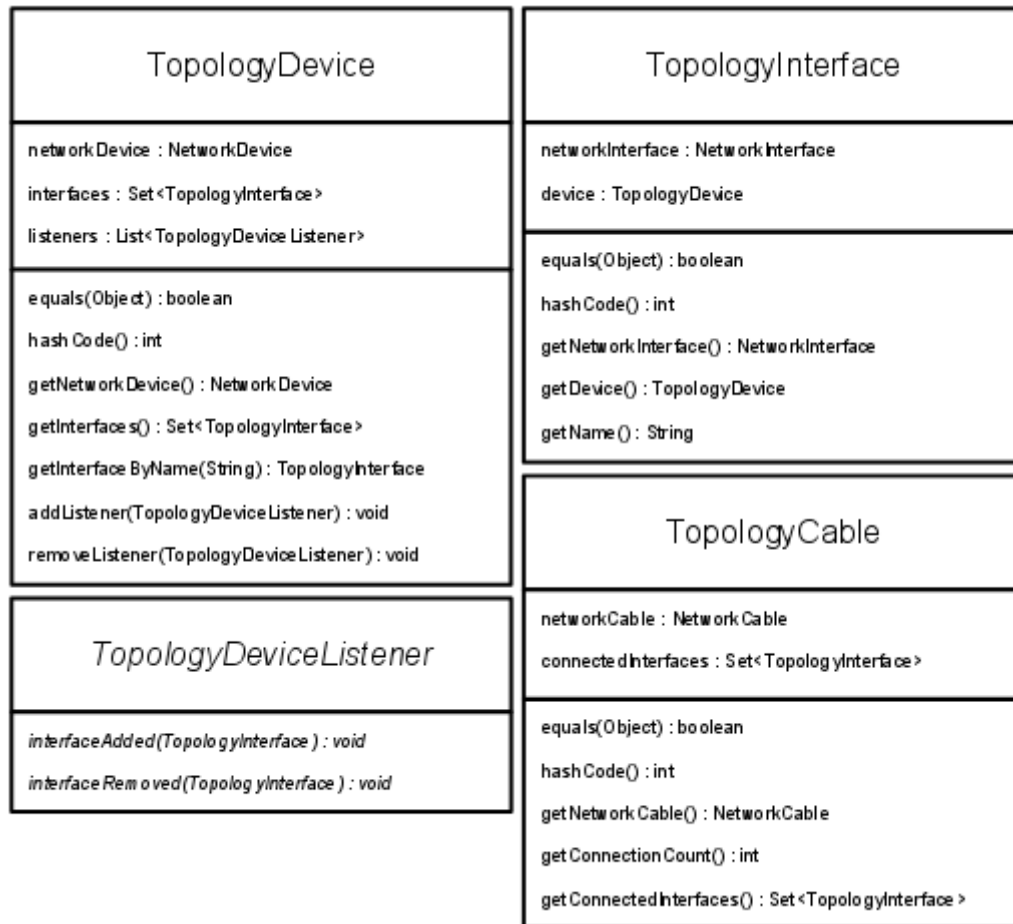


Abbildung 22: Die Klassen TopologyDevice, TopologyDeviceListener, TopologyInterface und TopologyCable in einer Übersicht

Wir modellierten zuerst die Klasse TopologyDevice. Sie soll ein Netzwerkgerät in unserer Topologie darstellen. Dazu erhielt sie in erster Linie die Attribute networkDevice und interfaces. Ersteres enthält alle Informationen zu diesem Gerät. Das Attribut interfaces beinhaltet die angeschlossenen Netzwerkschnittstellen. Die wichtigste Methode ist getInterfaceByName(). Sie greift auf getName() der einzelnen Schnittstellen zu, und sucht jenes mit dem angegebenen Namen.

Zum Erzeugen eines Objekts von TopologyDevice benötigt man eine Instanz der Klasse NetworkDevice. In Folge dessen wird eine Instanz von

NetworkModelListener erstellt, die dafür sorgt, dass für jedes NetworkInterface ein TopologyInterface erstellt wird.

Außerdem wurde die Schnittstelle TopologyDeviceListener erstellt um auf Änderungen der Netzwerkschnittstellen reagieren zu können.

Die Klasse TopologyInterface stellt eine Netzwerkschnittstelle in der Netzwerktopologie da. Sie hat die Attribute networkInterface, das alle Informationen der Schnittstelle enthält, und device, das auf das zugehörige Netzwerkgerät verweist. Die wichtigste Methode ist getName(). Sie greift über networkInterface auf den Namen der Netzwerkschnittstelle zu, und gibt diesen zurück.

Zuletzt wurde die Klasse TopologyCable erstellt. Sie stellt ein Netzkabel in der Topologie dar und enthält alle Informationen darüber im Attribut networkCable. Außerdem besitzt es eine Menge von allen verbundenen Netzwerkschnittstellen. Diese Information wird im Attribut connectedInterfaces abgelegt.

Nachdem alle Komponenten einer Netzwerktopologie in unserem Programm abgebildet waren, konnte schlussendlich selbige modelliert werden. Wir entschlossen uns dazu, eine Schnittstelle zu verwenden, da es verschiedene Implementierungen eines Graphen geben kann.

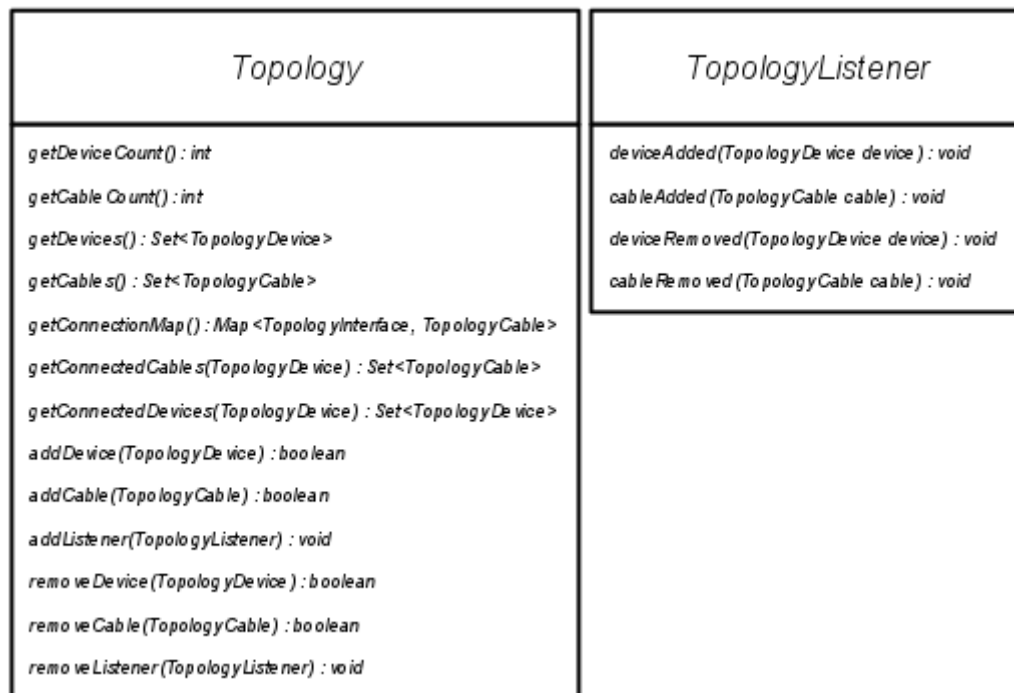


Abbildung 23: Die Klassen *Topology* und *TopologyListener* in einer Übersicht

Der Graph hat genau genommen nur eine Funktion: das Verbinden von Knoten mit Kanten, oder in unserem Fall, von Netzwerkgeräten mit Netzwerkkabel. Es ist also möglich Geräte hinzuzufügen und sie anschließend mit einem Kabel zu verbinden. Diese Funktion stellen die Methoden `addDevice` und `addCable` bereit. Dabei wird darauf geachtet ob eine Netzwerkschnittstelle schon in Verwendung ist, falls ja, wird die Operation abgebrochen.

Die Methoden `removeDevice` und `removeCable` entfernen wiederum ein Gerät beziehungsweise ein Kabel. Beim Löschen eines Gerätes ist zu beachten, dass automatisch alle verbundenen Kabel ebenfalls entfernt werden.

Um nun eine einfache Topologie mit zwei Geräten und einer Verbindung zusammenzustellen, reichen folgende Zeilen:

```

Topology topology = new HashTopology();

NetworkDevice deviceA = new NetworkDevice();
NetworkInterface interfaceA = new NetworkInterface();
interfaceA.setValue(NetworkInterface.NAME, "eth0");
deviceA.setValue(NetworkDevice.IDENTIFICATION, "a");
deviceA.setValue(NetworkDevice.HOSTNAME, "RouterA");
deviceA.setValue(NetworkDevice.INTERFACES,
    new
    HashSet<NetworkInterface>(Arrays.asList(interfaceA)));

NetworkDevice deviceB = new NetworkDevice();
NetworkInterface interfaceB = new NetworkInterface();
interfaceB.setValue(NetworkInterface.NAME, "eth0");
deviceB.setValue(NetworkDevice.IDENTIFICATION, "b");
deviceB.setValue(NetworkDevice.HOSTNAME, "RouterB");
deviceB.setValue(NetworkDevice.INTERFACES,
    new
    HashSet<NetworkInterface>(Arrays.asList(interfaceB)));

NetworkCable cable = new NetworkCable();

TopologyDevice topologyDeviceA = new TopologyDevice(deviceA);
TopologyInterface topologyInterfaceA =
    topologyDeviceA.getInterfaceByName("eth0");
TopologyDevice topologyDeviceB = new TopologyDevice(deviceB);
TopologyInterface topologyInterfaceB =
    topologyDeviceB.getInterfaceByName("eth0");
TopologyCable topologyCable = new TopologyCable(cable,
    new HashSet<TopologyInterface>(Arrays.asList(
        topologyInterfaceA,
        topologyInterfaceB)));

topology.addVertex(topologyDeviceA);
topology.addVertex(topologyDeviceB);
topology.addEdge(topologyCable);

```

7.8 Darstellung des Netzwerkes

Die Darstellung des Netzwerkes ist für den Benutzer wohl die wichtigste Funktion. Sie soll alle Netzwerkgeräte und alle Netzwerkverbindungen darstellen. Außerdem sollte es möglich sein, dass der Benutzer das Netzwerk nach eigenen Vorstellungen ausrichten kann. Dazu haben wir uns das “Model-View-Controller”-Konzept als Vorlage genommen. Model und Controller werden dabei durch die Topology Klasse abgedeckt. Die View wird durch die TopologyView ersetzt.

Ein Objekt der Klasse TopologyView generiert automatisch grafische Elemente für jedes neue Netzwerkgerät und Netzworkkabel. Das macht es möglich, während des Suchlaufs ein Bild des bereits erfassten Netzwerkes zu sehen.

7.9 Datenablage - TAS

Um Daten, wie zum Beispiel einen Suchlauf oder eine Konfiguration aus unserem Configuration Assistant, dauerhaft abzuspeichern, haben wir uns einer Spezialität einer anderen Programmiersprache bedient. Denn das von uns verwendete Format namens JSON, die JavaScript Object Notation, kommt, wie der Name schon sagt, von JavaScript. Es wurde von Douglas Crockford, einem wichtigen Mitglied der JavaScript-Gemeinde, erfunden und in einer späteren Version der Sprache hinzugefügt. Mittlerweile hat JSON andere Formate, wie XML, im so genannten “Web 2.0” von der Bildfläche fast vollständig verschwinden lassen, da mittlerweile einige zentrale Dienste, darunter auch Facebook, ihre Daten nur noch in diesem Format anderen Entwicklern zur Verfügung stellen.

Im Gegensatz zu anderen Formaten wie XML werden bei JSON die Objekte selbst in Form von Text dargestellt, anstatt die darin enthaltenen Daten in einer anderen Struktur als der des eigentlichen Objektes zu notieren. Die dafür verwendeten Elemente sind

überschaubar, denn im Endeffekt werden sämtliche Daten mithilfe einiger weniger Datentypen dargestellt. Die wichtigsten davon sind Zahlen, Strings, Listen und Maps, wobei Strings eine Anreihung von Buchstaben sind und Maps eine Art Wörterbuch, das einen Wert auf einen anderen auflöst. Zusätzlich zu diesen Werten gibt es noch Ausnahmen, die jedoch ebenfalls als Strings dargestellt werden könnten und in diesem Kapitel keine weitere Bedeutung haben.

Um das eben erklärte etwas zu veranschaulichen, wird folgend ein sehr einfaches Beispiel angeführt:

```
{
  "text": "abc",
  "list": ["a", "b", "c"],
  "map": {
    "eins": "1",
    "zwei": "2",
  },
}
```

Eine für Java-Programme naheliegende Alternative zu JSON wäre die Java Serialization API gewesen. Diese bildet ein ganzes Objekt in eine für Menschen unlesbare Zeichenkette ab. Der Vorteil gegenüber JSON wäre, dass das Resultat einer Serialization kürzer, dafür aber eben unlesbar ist. Da eine lesbare Ausgabe die Arbeit mit den Daten für uns vereinfacht, haben wir uns gegen die Java Serialization API und für JSON entschieden. Ein weiterer Grund, der für JSON gesprochen hat, ist, dass es ein passendes Format für eine spätere Schnittstelle für Dritte ist, da mittlerweile jede Programmiersprache damit umgehen kann.

7.10 Schnittstelle für Dritte - TAS

Eine Schnittstelle für Dritte ist insbesondere wichtig für ein Programm, das Daten sammelt oder erzeugt, um anderen Entwicklern zu erlauben

auf dem eigentlichen Programm aufzubauen, ohne es selbst zu verändern. Somit können zum Beispiel Firmen netCrawler um Ihre eigenen Visualisierungen erweitern, ohne unseren Code jemals verändern zu müssen. Da wir uns entschieden haben unser Projekt als Open-Source zu veröffentlichen, könnten natürlich auch entsprechende Änderungen an unserem Code vorgenommen werden. Das würde jedoch bedeuten, dass sich die interessierten Entwickler erst mit unserer Arbeit vertraut machen müssten, was oft ein langwieriger und anstrengender Prozess sein kann. Eine umfangreiche und flexible Schnittstelle für Dritte erlaubt es ihnen jedoch mit vergleichsweise wenig Arbeit mithilfe unserer Dokumentation netCrawler um gewünschte Funktionen zu erweitern.

Für die vorläufige Version unseres Projektes haben wir uns entschieden, keine umfangreiche Schnittstelle für Dritte bereitzustellen. Zuvor soll das Feedback von Benutzern dazu verwendet werden, die am häufigsten gebrauchten Funktionen herauszufinden, auszubauen und gegebenenfalls gänzlich neue Funktionen hinzuzufügen.

Daher gibt es zur Zeit ausschließlich zwei Möglichkeiten, einen Suchlauf zu starten und mit den gesammelten Daten zu interagieren. Eine davon ist die grafische Oberfläche, die in den Kapiteln “8. Benutzerhandbuch” und “7.11 Grafische Oberfläche” genauer beschrieben wird. Für andere Einsatzgebiete, in denen keine grafischen Oberflächen möglich sind, wie zum Beispiel auf einem so genannten “headless server”, kann netCrawler auch über die Kommandozeile gestartet werden. Der Suchlauf kann somit gestartet und nach Beendigung automatisch abgespeichert werden. Auf die daraus resultierende .crawl-Datei können Dritte zugreifen und damit machen was sie wollen.

Die einfachste Möglichkeit einen Suchlauf über die Kommandozeile zu starten lautet wie folgt:

```
java -jar netcrawler.jar --crawl -s <IP-Adresse> -u <USER> -p  
<PASSWORD>
```

Der Parameter `crawl` bestimmt die Aktion, die durchgeführt werden soll, `s` bei welcher IP der Suchlauf gestartet werden soll, und `u` und `p` legen den zu verwendenden Benutzernamen und das Passwort fest.

7.11 Grafische Oberfläche - TAS

Bei der Erstellung der grafischen Oberfläche stellte sich für uns die Frage, wie wir so viele verschiedene Daten und Funktionen in eine zusammenhängende und intuitive Arbeitsumgebung verpacken können. Das Resultat unserer Überlegungen ähnelt früheren Versionen von `netCrawler`, unterscheidet sich aber, abgesehen von dem Mehrwert an Funktionen, durch die Möglichkeit die Ansicht zu ändern, mit der man die gefundenen Daten betrachtet. So ist es nun möglich, zusätzlich zu der bereits bekannten grafischen Visualisierung, die Geräte und deren wichtigste Informationen in Form einer Liste anzeigen zu lassen. Das hat den Vorteil, dass hier auch Informationen wie IP-Adressen und Laufzeit des Gerätes ohne weitere Klicks eingesehen werden können. Wie bereits angesprochen, waren in früheren Versionen von `netCrawler` beide Ansichten gleichzeitig eingeblendet, was die Oberfläche aufgeblasen wirken hat lassen.

Die bereits erwähnte grafische Visualisierung der Geräte wird von unserem `TopologyViewer` erstellt, welcher im folgenden Kapitel "7.8 Darstellung des Netzwerkes" genauer beschrieben wird.

Eine weiterer wichtiger Meilenstein, der die Verwendung unserer Projekte stark vereinfacht hat, war die Verschmelzung unseres `Configuration Assistant` mit `netCrawler`. Zuvor waren diese Produkte

unabhängig voneinander zu bedienen, mittlerweile kann der Configuration Assistant aber direkt aus netCrawler heraus gestartet werden. Die genaue Funktionsweise und die daraus resultierenden Vorteile können im Kapitel “8 Benutzerhandbuch” nachgelesen werden.

8 BENUTZERHANDBUCH - KER

8.1 Oberfläche von netCrawler

8.1.1 Geräteansicht

Der Hauptteil der grafischen Oberfläche ist für die Geräteansicht bestimmt. Die Standardansicht ist die grafische Darstellung des Netzwerkes.



Abbildung 24: Die Oberfläche von netCrawler, wie sie nach dem Starten aussieht

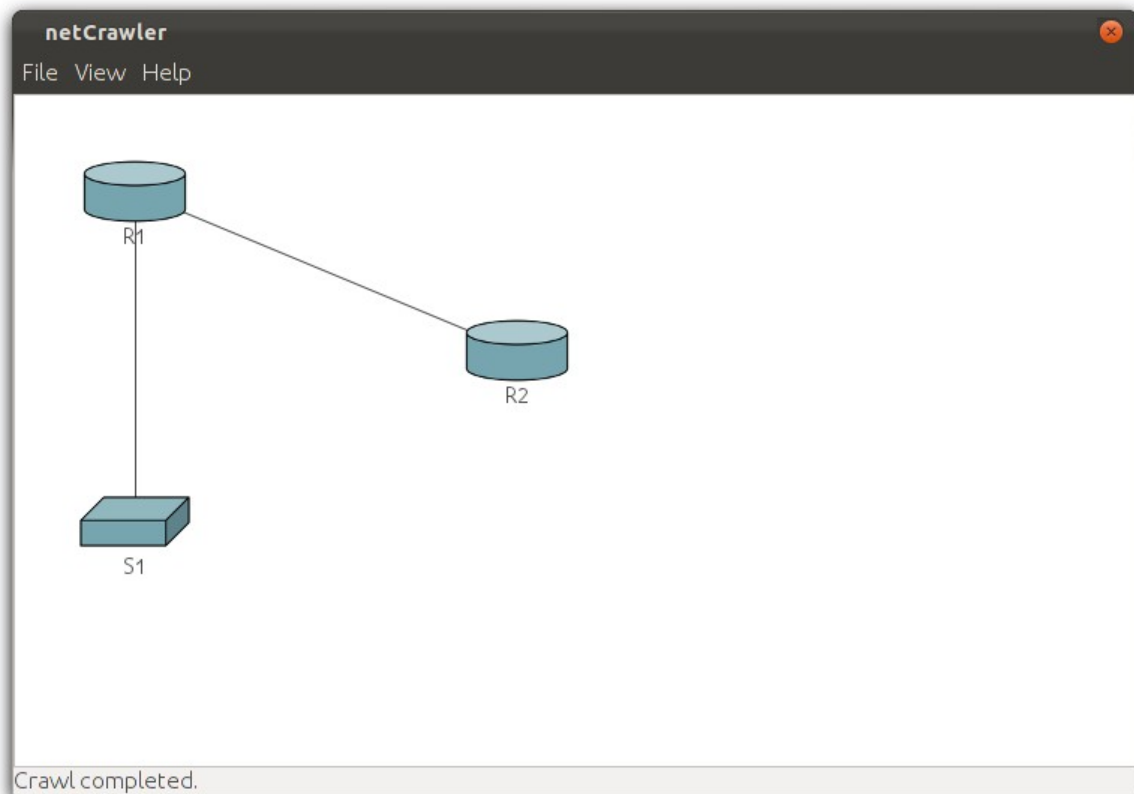
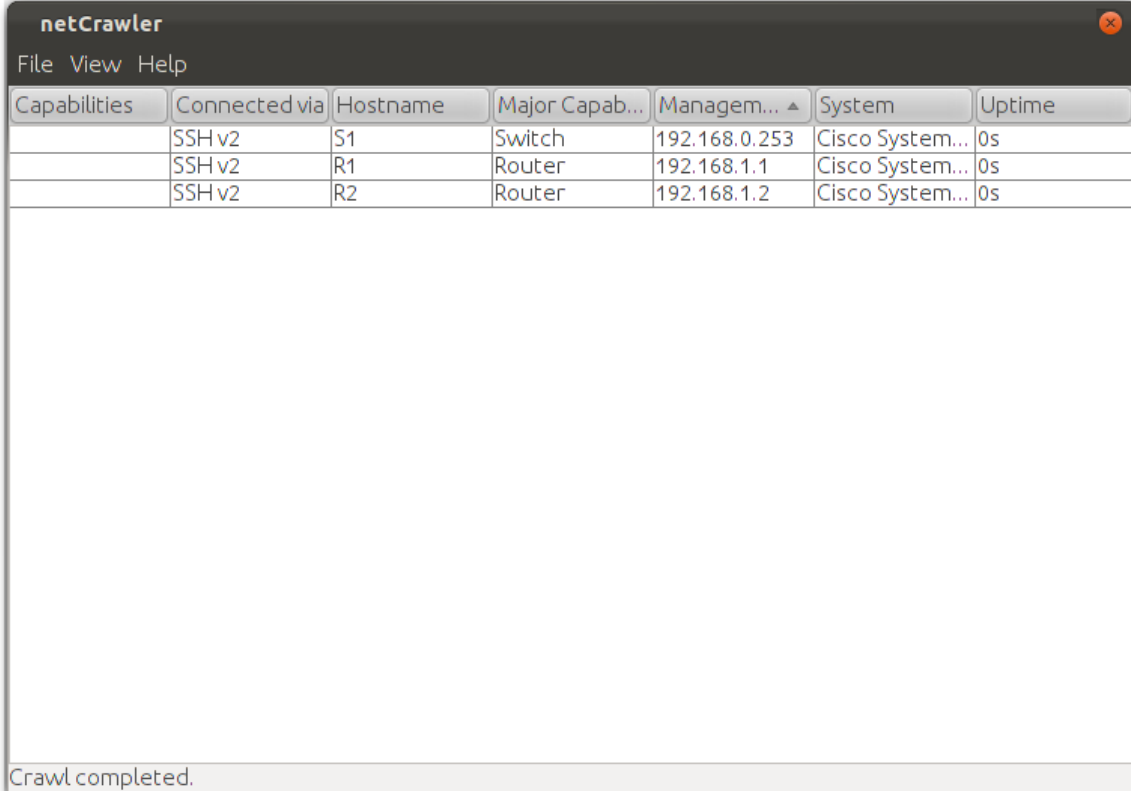


Abbildung 25: Die Oberfläche von netCrawler, wie sie nach einem Suchlauf aussieht

Hier sieht man die Geräte des Netzes mit deren Hostnamen und den korrekten Verbindungen. In dieser Ansicht können die Geräte auch beliebig verschoben werden, da die Ausrichtung der Geräte nach einem Suchlauf etwas unübersichtlich sein kann.

Außerdem erkennt man den Status von netCrawler in der Statusleiste am unteren Rand der Oberfläche. Hier steht entweder, dass auf einen Suchlauf gewartet wird, oder dass gerade gesucht wird.

Man kann die Ansicht auch über den Menüpunkt "View" - "Toggle table/topology view" auf die Listenansicht umstellen. Diese sieht folgendermaßen aus:



The screenshot shows a window titled "netCrawler" with a menu bar containing "File", "View", and "Help". Below the menu bar is a table with the following columns: "Capabilities", "Connected via", "Hostname", "Major Capab...", "Managem...", "System", and "Uptime". The table contains three rows of data:

Capabilities	Connected via	Hostname	Major Capab...	Managem...	System	Uptime
	SSH v2	S1	Switch	192.168.0.253	Cisco System...	0s
	SSH v2	R1	Router	192.168.1.1	Cisco System...	0s
	SSH v2	R2	Router	192.168.1.2	Cisco System...	0s

Below the table, the status "Crawl completed." is displayed.

Abbildung 26: Die zuvor gecrawlte Topologie in Form einer Liste

Grundsätzlich ist diese Ansicht mit den gleichen Funktionen bestückt wie die Topologieansicht. Hier reicht, wie in der Topologieansicht, ein Klick auf einen Gerätenamen um die Gerätansicht zu öffnen. Außerdem können die Geräte durch einen Klick auf die Spaltenüberschriften entsprechend sortiert werden. Mit einem Rechtslick auf das gewünschte Gerät, öffnet sich der Configuration Manager, ein Teil unseres Configuration Assistants. In diesem können Konfigurationen erstellt werden, welche später mithilfe des Configuration Executors ausgeführt werden können. Mit einem Klick auf "Execute" wird der Batch Executor geöffnet, von welchem aus die zuvor erstellten Batches ausgeführt werden.

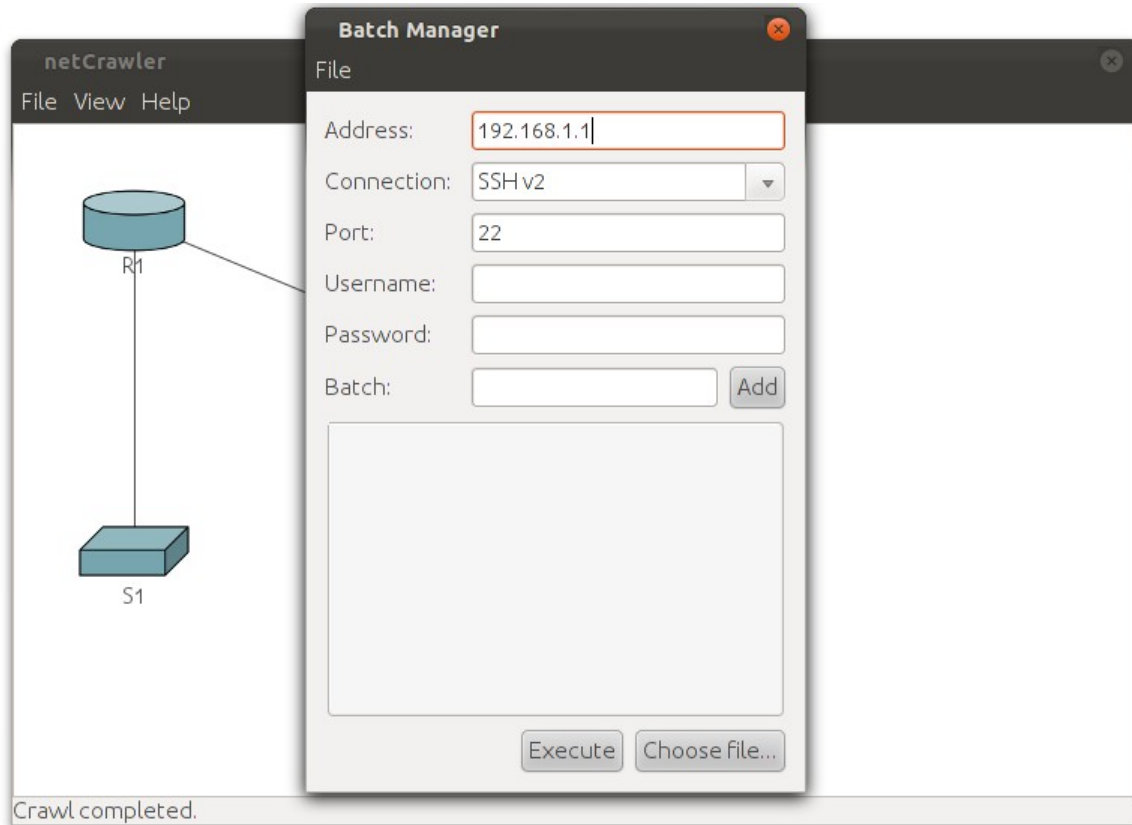


Abbildung 27: Der Batch Manager mit automatisch eingetragenen Verbindungsdaten

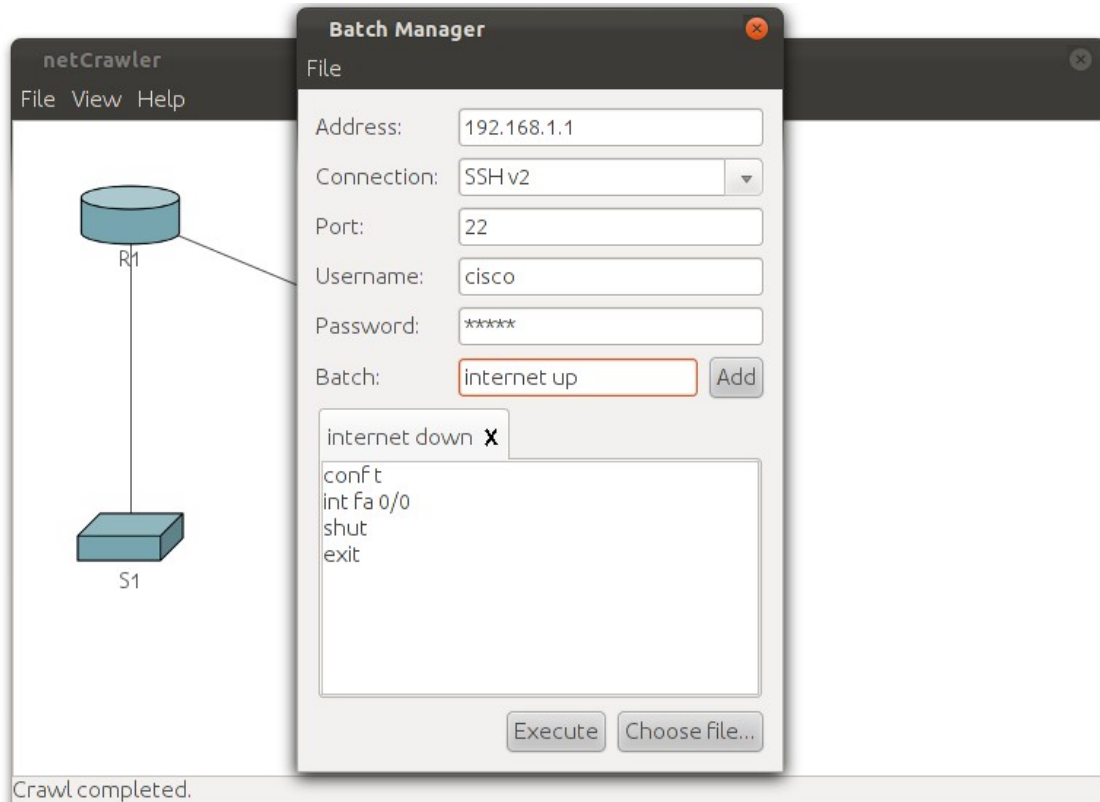


Abbildung 28: Der Batch Manager mit einer möglichen Batch um die Netzwerkkonnektivität zu kappen

Durch einen Klick auf ein Gerät, sowohl in der grafischen Darstellung, als auch in der Listenansicht, öffnet sich die spezifische Geräteansicht. Hier können die wichtigsten Informationen des Gerätes ausgelesen und das Gerät konfiguriert werden.

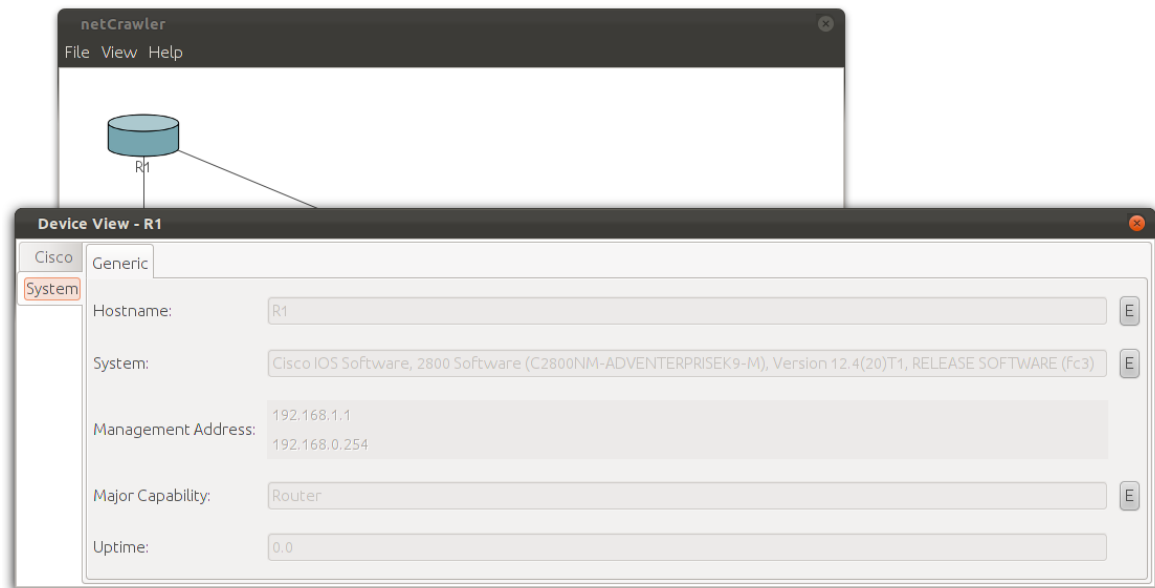


Abbildung 29: Die Geräte-Ansicht

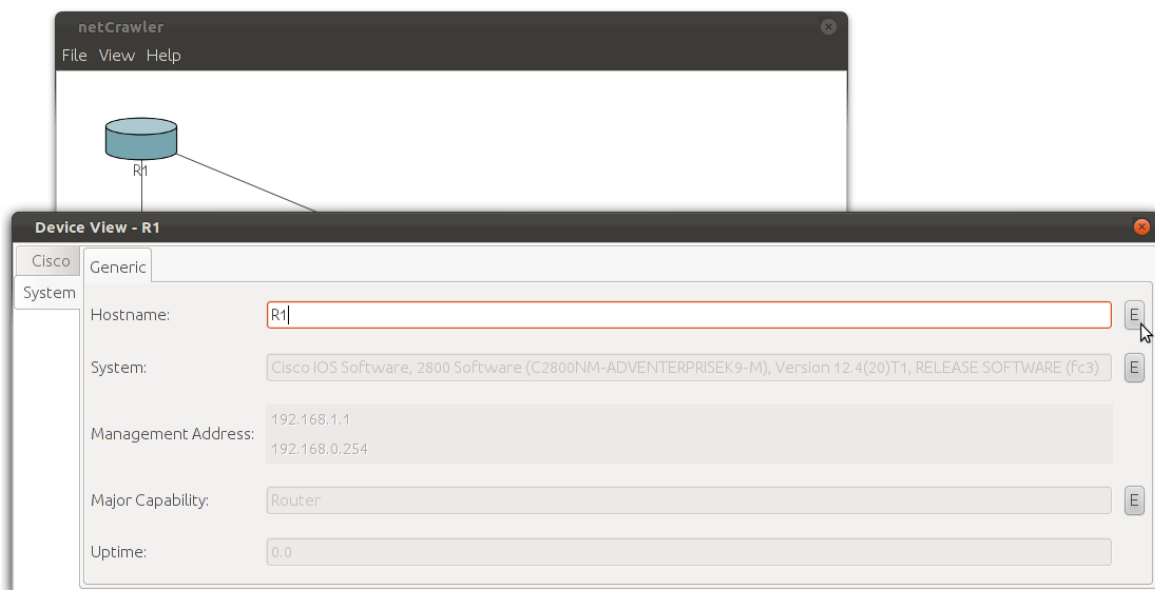


Abbildung 30: Die Geräte-Ansicht im Editier-Modus

8.1.2 Menü

Im Menü von netCrawler gibt es 3 wichtige Elemente, nämlich: File, View und Help.

Im Menüpunkt "File" kann man einen neuen Suchlauf starten, indem man den Unterpunkt "Crawl" wählt. Außerdem ist es möglich den

letzten Suchlauf mit "Save" zu speichern und mit "Load" wieder zu laden.

Der Menüpunkt "View" wird im Kapitel "8.1.1 Geräteansicht" erklärt.

Über "Help" gelangt man zu unserer weiterführenden Online-Hilfe, welche sowohl eine kurze Auflistung wichtiger Fragen, als auch eine detaillierte Hilfestellung bietet.

8.2 Ändern von Gerätekonfigurationen

Die Geräte werden, wie unter Punkt "8.1.1 Geräteansicht" bereits kurz erwähnt, über die spezifische Geräteansicht konfiguriert.

Auf der linken Seite der Oberfläche befindet sich eine Liste mit Kategorien der wichtigsten, ausgelesenen Informationen. Mit einem Klick auf ein Element der Liste öffnet sich im rechten Teil des Fensters eine Ansicht mit weiterführenden Tabs, welche die Daten dieser Kategorie weiter untergliedern.

In diesen Tabs kann nun die Gerätekonfiguration betrachtet und auch geändert werden.

Im Menü gibt es außerdem die Punkte "SNMP" und "Console". Durch Wahl von "SNMP" öffnet sich ein Fenster, in welchem es möglich ist, gezielt Daten über deren OID vom Gerät auszulesen, oder, wenn möglich, am Gerät zu ändern. Die Informationen beziehungsweise das Resultat der Konfigurationsänderung wird im Textfeld unterhalb angezeigt.

Durch einen Klick auf "Console" öffnet sich eine Kommandozeile, um die Informationen mittels SSH oder Telnet auszulesen oder das Gerät manuell zu konfigurieren.

8.2.1 Konfigurieren eines Gerätes

Wie bereits erwähnt wird die Konfiguration der Geräte in der spezifischen Geräteansicht vorgenommen. Durch einen Klick auf den gewünschten Tab erhält man die Informationen des Gerätes. Jede Information, welche geändert werden kann, hat ein Textfeld, welches zu Beginn nicht editierbar ist. Erst durch einen Klick auf den Editier-Button auf der rechten Seite, kann der darin befindliche Wert geändert werden. Um die Änderungen an das Gerät zu senden muss erneut auf den Button geklickt werden.

8.2.2 Konfigurieren mehrere Geräte

Der Batch Manager, dient dazu mehrere Geräte auf einmal zu konfigurieren. Es müssen lediglich die IP-Adressen der Geräte im Adressfeld angegeben werden, zum Beispiel wie folgt: "192.168.0.1;192.168.0.2;192.168.0.2". Durch einen Klick auf den "Execute"-Button, wird der Batch Executor geöffnet, welcher die Batches auf allen Geräten gleichzeitig ausführt.

9 ERGEBNIS - TAS

9.1 *netCrawler*

netCrawler war das ursprüngliche Produkt an dem wir in dieser Diplomarbeit gearbeitet haben. Außerdem basieren die meisten Nebenprodukte, die nachfolgend beschrieben werden, auf dem Code, der für netCrawler geschrieben wurde.

netCrawler erkennt während eines einmaligen Suchlaufes die intermediären Geräte in einem Netzwerk, wie Router, Switch und Firewall, und visualisiert diese in Folge, je nach Bedarf auf verschiedene Weisen. Anschließend ist es über diese Visualisierungen möglich, einzelne oder auch mehrere Geräte gleichzeitig zu konfigurieren. Somit sollen besonders Netzwerkadministratoren oft anfallende Arbeiten erleichtert werden.

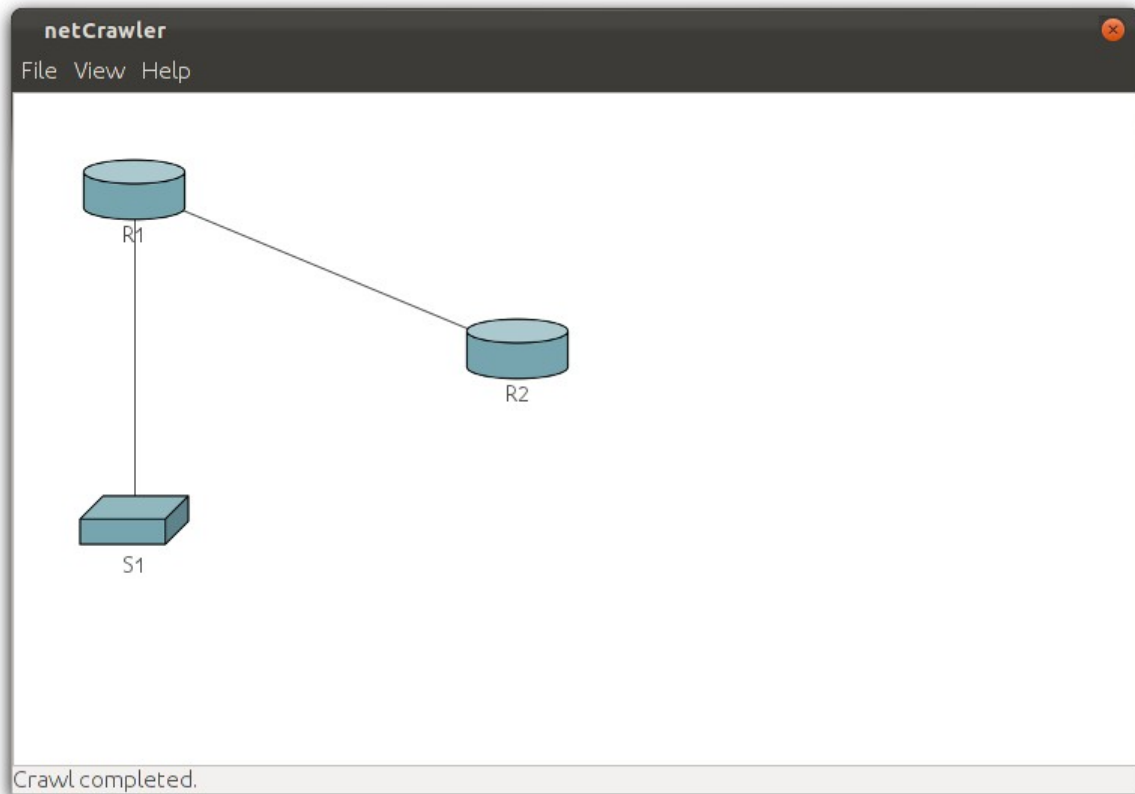


Abbildung 31: Die Oberfläche von netCrawler, wie sie nach einem Suchlauf aussieht

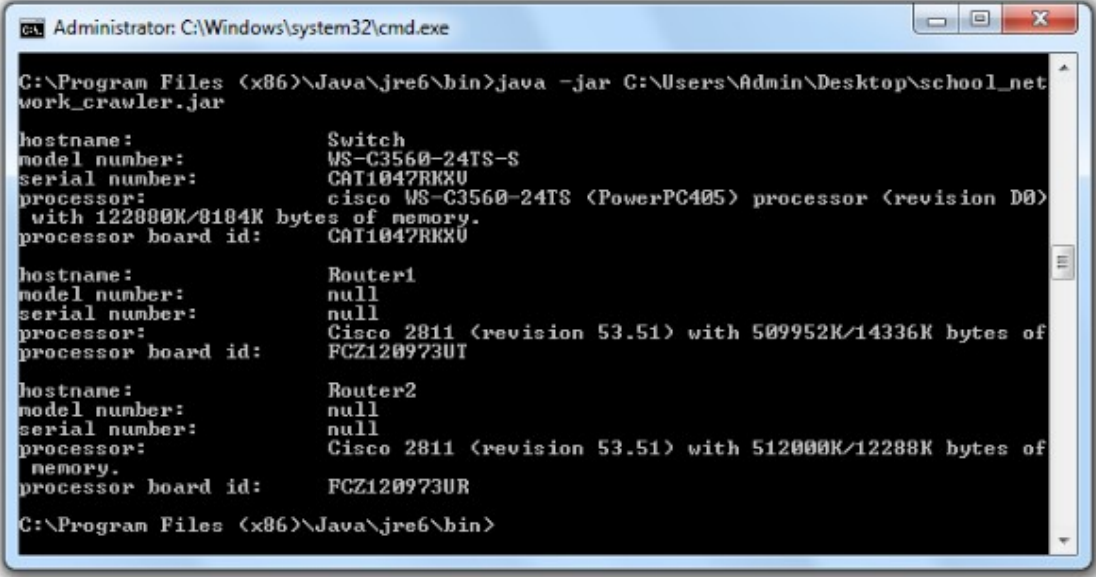
9.2 schoolCrawler

schoolCrawler war ein erster Meilenstein in der Entwicklung von netCrawler. Um die in unserer Schule verwendeten Netzwerkgeräte für eine anstehende Vertragsverlängerung zu inventarisieren, hat uns Herr Professor Schöndorfer darum gebeten eine stark vereinfachte Version von netCrawler für diesen Zweck zu erstellen. Diese Version basierte bereits auf dem selben Code, der heute in netCrawler arbeitet.

Da schoolCrawler nur für diesen einen Zweck gedacht war, wurde auf jegliche grafische Oberfläche verzichtet. Das Programm wurde ausschließlich über die Kommandozeile bedient. Nach dem Starten eines Suchlaufes, wurden nach und nach die benötigten Informationen, unter anderem Hostname, Modell- und Seriennummer und der Typ des

Prozessors in der Konsole ausgegeben. Bei Bedarf wurde nach Benutzernamen und Passwort gefragt, um sich auf ein bestimmtes Gerät zu verbinden.

Falls andere Administratoren ebenfalls vor der Aufgabe stehen, ein umfangreicheres Netzwerk zu inventarisieren, kann schoolCrawler ohne weitere Änderungen verwendet werden, falls die gleichen Informationen wie in unserem Fall benötigt werden. Anderenfalls ist es ohne großem Aufwand möglich mehr oder weniger Informationen ausgeben zu lassen. Außerdem muss das Netzwerk, das durchsucht werden soll, ausschließlich aus Cisco Geräten bestehen, damit schoolCrawler sämtliche Geräte findet.



```
Administrator: C:\Windows\system32\cmd.exe

C:\Program Files (x86)\Java\jre6\bin>java -jar C:\Users\Admin\Desktop\school_network_crawler.jar

hostname:          Switch
model number:      WS-C3560-24TS-S
serial number:     CAT1047RKXU
processor:         cisco WS-C3560-24TS (PowerPC405) processor (revision D0)
                  with 122800K/8104K bytes of memory.
processor board id: CAT1047RKXU

hostname:          Router1
model number:      null
serial number:     null
processor:         Cisco 2811 (revision 53.51) with 509952K/14336K bytes of
processor board id: FCZ120973UT

hostname:          Router2
model number:      null
serial number:     null
processor:         Cisco 2811 (revision 53.51) with 512000K/12288K bytes of
memory.
processor board id: FCZ120973UR

C:\Program Files (x86)\Java\jre6\bin>
```

Abbildung 32: Die Ausgabe eines Suchlaufes mit schoolCrawler

9.3 Packet Tracer Bridge

Die Packet Tracer Bridge ist vorrangig ein Testwerkzeug, welches uns erlaubt von einem realen Netzwerk auf das Netzwerk von Packet Tracer zuzugreifen. Dadurch ist es uns möglich, netCrawler zu testen, ohne ein

reales Netzwerk aufzubauen, was finanziell für uns nicht möglich gewesen wäre. Außerdem hätten wir mehrere physikalische Netzwerke erwerben müssen, damit jeder Mitarbeiter jederzeit zuhause an netCrawler arbeiten kann.

Zurzeit unterstützt die Packet Tracer Bridge Telnet, da es in der Testphase für netCrawler benötigt wurde. Spätere Versionen sollen es aber erlauben, auch mit Protokollen wie HTTP und DNS zu arbeiten.

9.4 Configuration Assistant

Der Configuration Assistant dient hauptsächlich dazu, Befugten ohne weitere Kenntnisse zu ermöglichen, vordefinierte Befehle auf einem Netzwerkgerät auszuführen. Der Bedarf, den wir zu erfüllen versuchten war, einem Lehrer während eines Tests den Internetzugang für Schüler zu sperren. Nach dem Test sollte dieser den Internetzugang natürlich auch wieder freigeben können.

Normalerweise müsste in so einem Fall der Lehrer den Netzwerkadministrator bitten, den Internetzugang zu sperren, da im besten Falle nur der Administrator die Konfiguration der Geräte ändern darf. Je nach Größe der Schule könnte diese Aufgabe jedoch mehrmals pro Tag von Nöten sein, was einen immensen Aufwand für den Administrator bedeuten würde.

Der Configuration Assistant besteht aus zwei Komponenten. Einerseits gibt es den Configuration Manager, der ausschließlich für den Administrator gedacht ist, andererseits gibt es den Configuration Executor, der für denjenigen gedacht ist, der die Konfiguration eines Administrators ausführen will.

Mithilfe des Configuration Managers ist es nun möglich, dass der Administrator lediglich ein einziges Mal die nötigen Befehle niederschreibt, um den Internetzugang in einem bestimmten Raum zu sperren. Diese Befehle übergibt er samt benötigtem Benutzernamen, Passwort und IP-Adresse dem Configuration Manager. Dieser erstellt daraus eine Konfiguration, welche mit dem Configuration Executor von einem Lehrer ohne jegliche Kenntnisse ausgeführt werden kann. Um Benutzernamen und Passwort nicht an interessierte Schüler zu verraten, kann eine Konfiguration auch verschlüsselt werden.

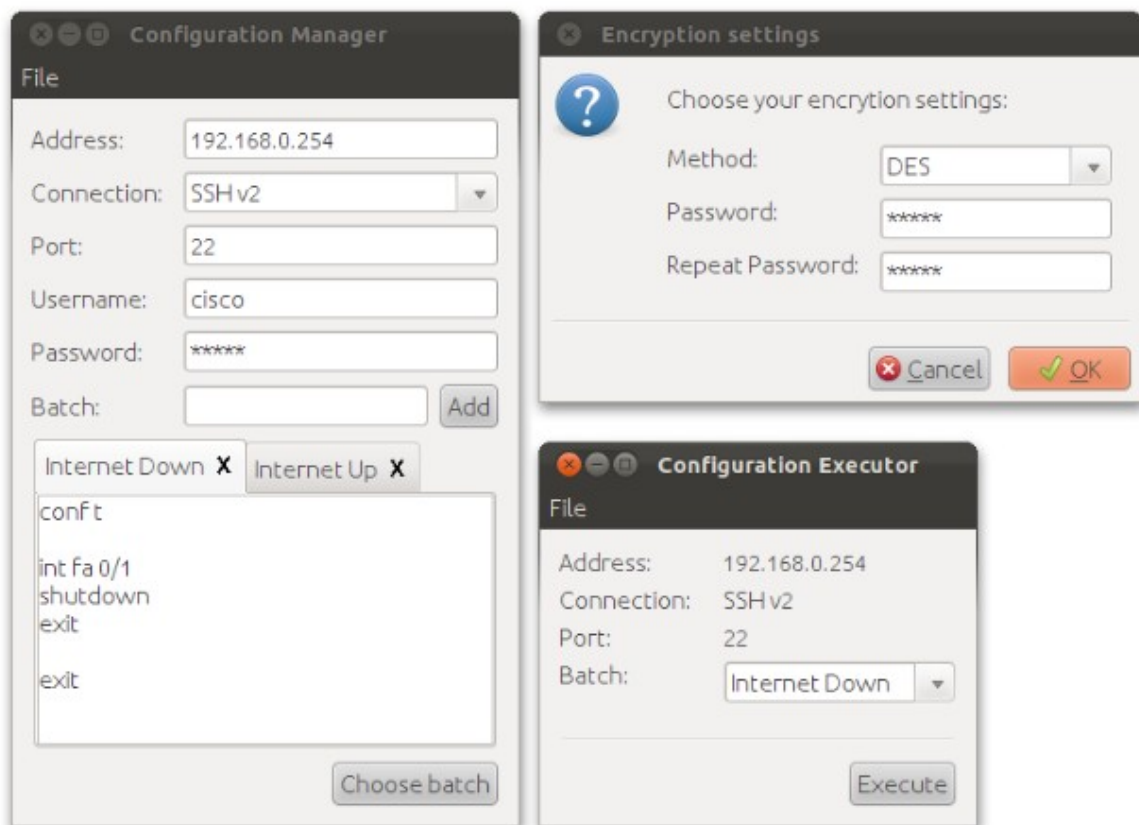


Abbildung 33: Der Configuration Manager zum Erstellen von Konfigurationen, der Configuration Executor zum Ausführen dieser

10 FAZIT

10.1 Persönliche Erfahrungen

10.1.1 Thomas Taschauer

Ich habe es im Laufe des letzten Jahres kein einziges Mal bereut mich für eine Diplomarbeit entschieden zu haben, da ich im Zuge dessen viele verschiedene Erfahrungen gemacht habe. Zum einen habe ich aus Fehlern lernen können, andererseits habe ich aber auch spannende Erfahrungen, wie das Auftreten vor einem größeren Publikum, gemacht.

Neben meinen Erfahrungen als Projektleiter habe ich außerdem viele Erfahrungen mit den von uns verwendeten Protokollen gemacht, und auch Neues bei der Entwicklung des Programmes kennengelernt.

Eine für mich als Projektleiter sehr wichtige Erkenntnis ist, dass die Planung von Anfang bis Ende am neusten Stand gehalten werden sollte, und keinesfalls vernachlässigt werden darf. Außerdem muss darauf geachtet werden, dass sämtliche Mitarbeiter kontinuierlich arbeiten, um Stress vor wichtigen Terminen zu vermeiden.

10.1.2 Andreas Stefl

Die Diplomarbeit war für mich persönlich eine große Herausforderung, da sie einen sehr weiten Umfang hatte. Dies machte es jedoch umso interessanter und spannender. Durch den Umfang wurde das Projekt nämlich in mehrere kleine Module zerlegt, die einzeln abgearbeitet und zu gewissen Meilensteinen zusammengefasst wurden. Die Spannung lag darin, dass alle Module letztendlich, wie die Zahnräder in einem Uhrwerk, ineinander griffen.

Außerdem war es auch neu für mich, ein Programm als Team zu entwickeln, was viele Vorteile und auch Nachteile mit sich bringt. Einerseits kommt man sehr schnell voran und kann sich gegenseitig über Verbesserungsvorschläge austauschen. Andererseits kann es auch zu Konflikten führen, wenn die Kommunikation nicht gut abgestimmt ist. Letzteres ist jedoch sehr selten passiert und wurde auch schnell wieder bereinigt.

Alles in allem freue ich mich rückblickend sehr darüber, dass ich mich für eine Diplomarbeit entschieden habe. Wie schon erwähnt war es eine Herausforderung, die wir gemeinsam bewältigen, und dabei einiges an Erfahrung sammeln konnten.

10.1.3 Daniel Kern

Ich bin sehr froh, dass ich mich für eine Diplomarbeit entschieden habe, da ich dadurch gelernt habe, mit meiner Zeit besser um zu gehen. Mir wurde auch bewusst, dass eine konkrete Planung bei einem so langem Projekt sehr wichtig ist, da man sonst alle Arbeiten auf einmal erledigen muss.

Des weiteren lernte ich, eine große Menge an Arbeiten in einer gewissen Zeit zu erledigen, und zwar so, dass diese auch gewissenhaft ausgeführt wurden.

Natürlich lief nicht alles glatt, jedoch gab es keine großen Schwierigkeiten, was mich im Nachhinein sehr froh stimmt. Außerdem lernten wir aus unseren kleinen Fehlern und konnten diese in Zukunft verhindern, was uns sehr viel Zeit gespart hat.

Jedoch profitierte ich nicht nur von den Planungsmethoden und Techniken sondern auch von der netzwerktechnischen Materie. Ich

lernte im Laufe der Diplomarbeit viel über Netzwerktechnik und etwaige Protokolle.

Da ich auch für das Marketing zuständig war, stimmte es mich etwas traurig, dass so viele Firmen unsere Anfragen einfach ignorierten oder falsche Zusagen tätigten, indem sie eine Antwort versprachen, sich jedoch nicht mehr meldeten. Dieses Problem hatten unsere Kollegen ebenso und ich denke, man sollte junge Projekte fördern, da diese Menschen die Zukunft sind.

11 AUSBLICK - TAS

11.1 Wie geht es mit dem Projekt weiter?

Nach Abschluss dieser Diplomarbeit werden wir unsere Produkte in unserer Freizeit weiter verbessern und mögliche neue Funktionen in Betracht ziehen. Bei der Packet Tracer Bridge könnten zum Beispiel weitere Protokolle implementiert werden. Bei netCrawler wäre eine wichtige Funktion die Implementierung einer umfangreicheren Schnittstelle für Dritte, auf die bereits in einem der vorhergehenden Kapitel Bezug genommen wurde.

11.2 Gibt es Interessenten?

Wie sich bereits auf der AINAC gezeigt hat, hat unser Projekt Potenzial und weckt Interesse bei den verschiedensten Berufssparten. netCrawler selbst ist besonders für Netzwerkadministratoren interessant, während sowohl der Configuration Assistant als auch die Packet Tracer Bridge vielmehr das Interesse von Lehrern und IT-Kustoden an Schulen geweckt haben.

Anfangs werden unsere Produkte hauptsächlich in unserer Schule, der HTL Rennweg, verwendet werden, da wir bis jetzt kaum aufsehenerregendes Marketing betrieben haben. Später könnte sich die Zahl der Anwender durch Mundpropaganda und mögliche Werbeschaltungen stetig steigern.

12 QUELLENVERZEICHNIS

- NTMA-Unterlagen der 4. Klasse der HTL Rennweg von Thomas Angerer
- <http://de.wikipedia.org/wiki/Telnet> am 05.01.2012
- <http://de.wikipedia.org/wiki/Ssh> am 05.01.2012
- http://de.wikipedia.org/wiki/Cisco_Discovery_Protocol am 05.01.2012
- <http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en>
- http://de.wikipedia.org/wiki/Address_Resolution_Protocol am 05.01.2012
- http://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol am 05.01.2012
- <http://www.profinet.felser.ch/technik/LLDP.pdf>
- <http://de.wikipedia.org/wiki/LLDP> am 05.01.2012
- <http://www.networksorcery.com/enp/protocol/telnet.htm> am 21.01.2012
- <http://bauhaus.ece.curtin.edu.au/~iain/CTP392/PT%20development/PTMP.pdf> am 12.12.2011
- <https://sites.google.com/site/gson/gson-user-guide#TOC-Custom-Serialization-and-Deserializ> am 5.03.2012
- <http://help.github.com/create-a-repo/> am 10.10.2011