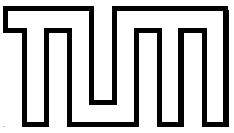


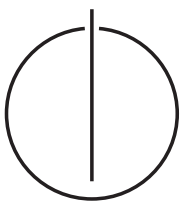
Technische Universität München

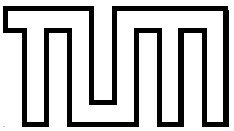
Fakultät für Informatik

Bachelorarbeit in Informatik

Intelligente Topologieerkennung zur Lokalisierung infizierter Rechner im Münchner Wissenschaftsnetz (MWN)

Steffen Wagner





Technische Universität München

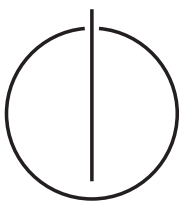
Fakultät für Informatik

Bachelorarbeit in Informatik

Intelligente Topologieerkennung zur Lokalisierung infizierter Rechner im Münchner Wissenschaftsnetz (MWN)

Intelligent Topology Recognition for Localization of Infected Computers within the Munich Scientific Network (MWN)

Bearbeiter:	Steffen Wagner
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dipl.-Inf. Ralf Kornberger Dr. Helmut Reiser Claus Wimmer
Abgabedatum:	12. September 2008



Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 12. September 2008

. . . . . .
(Unterschrift des Kandidaten)

Danksagung

Mein besonderer Dank gilt an dieser Stelle meinem Betreuer Dipl.-Inf. Ralf Kornberger, der mir während meiner Bachelorarbeit in allen Belangen mit Rat und Tat zur Seite stand.

Ich möchte mich auch bei Herrn Prof. Dr. Heinz-Gerd Hegering für die Ausschreibung des Themas bedanken. Ebenso gilt mein Dank Herrn Dr. Helmut Reiser für seine Unterstützung bei der Erstellung dieser Ausarbeitung.

In diesem Zusammenhang möchte ich auch den Lektoren Werner Hehl, Frank Höhn sowie Gerda und Wolfgang Klein danken, die freundlicherweise meine Arbeit gelesen und an den unterschiedlichsten Stellen verbessert haben.

Schließlich möchte ich mich auch bei meiner Familie – vor allem natürlich bei meinen Eltern, Karl-Heinz und Doris Wagner, sowie bei meinen Großeltern – ausdrücklich bedanken. Sie haben mich stets motiviert, mir den Rücken gestärkt und mir dieses Studium überhaupt erst ermöglicht.

Zusammenfassung

Diese Bachelorarbeit analysiert erstmals auf mathematisch fundierte Weise die Erkennungsleistung des Echtzeit-Lokalisierungssystems *Nyx*. Dazu werden verschiedene Ansätze diskutiert und ein eigens entwickeltes Skript zur automatisierten Analyse von Nyx präsentiert. Dieses liefert deutlich präzisere Ergebnisse als bisherige Ansätze, die teilweise lediglich auf Vermutungen und Schätzungen basierten. Die Herausforderung dabei war die Erarbeitung und Umsetzung der theoretischen Grundlagen. Auf den Ergebnissen der Analyse aufbauend werden dann der maschinelle Lernalgorithmus und dessen Trainingsdaten optimiert, um charakteristische Muster besser zu erkennen. Nyx verwendet nämlich Mustererkennung statt klassischer Topologieerkennung zur Lokalisierung von Endgeräten am Netzrand. Unter Lokalisierung versteht man in diesem Zusammenhang die Ermittlung derjenigen Netzkomponente, an der ein bestimmtes Endgerät angeschlossen ist, einschließlich des exakten Anschlussports. Dies ist besonders bei sehr großen und komplexen Kommunikationsnetzen keine triviale Aufgabe. Ein weiteres wichtiges Ergebnis dieser Arbeit ist deshalb die Steigerung der Erkennungsleistung durch die Verbesserung des maschinellen Lernverfahrens, was eine effektivere Lokalisierung zur Folge hat.

Abstract

This bachelor thesis analyzes the recognition accuracy of the real time localization system *Nyx* for the first time in a mathematically founded way. For this purpose different approaches are discussed and a script specially developed for automated analysis of Nyx is presented. That script reaches more precise results than previous approaches, which were merely based on assumptions and estimations. Thereby, the challenge was the development and realization of the theoretical bases. Subsequently the machine learning algorithm and its training data are optimized in order to recognize characteristic pattern more correctly. The reason is that Nyx uses pattern recognition instead of classical topology recognition to localize terminals at the edge of network. In this context localization is the determination of that network component, which a certain terminal is connected to, including the exact connection port. That task is not trivial – in particular not within very large and complex communications networks. So, an important result of this work is the increase of the recognition accuracy and a more effective localization by improvements of the machine learning mechanisms.

Inhaltsverzeichnis

1 Einführung

In sehr großen Kommunikationsnetzen mit mehreren zehntausend Benutzern ist die Ermittlung derjenigen Netzkomponente, an der ein bestimmtes Endgerät angeschlossen ist, keine triviale Aufgabe. Die Komplexität solcher Netze ist meist derart hoch, dass selbst ausgereifte Verfahren und Topologieerkennungsprotokolle an ihre Grenzen stoßen. Wesentliche Gründe für die hohe Komplexität sind die große Anzahl an Komponenten, die dem Aufbau des Netzes selbst dienen (Netzkomponenten), deren Heterogenität, die in der Praxis oft unvermeidbar ist, und die Vielzahl an Endgeräten, die an den Netzkomponenten angeschlossen sind. Die Gründe, warum eine Netzkomponente überhaupt ermittelt werden muss, sind ganz unterschiedlich und werden im Rahmen des Szenarios beispielhaft beschrieben.

Angesichts dieser komplexen Infrastruktur sieht sich der Betreiber eines solchen Kommunikationsnetzes scheinbar einem unlösbaren Problem gegenüber. Aus diesem Grund wurde für das „Münchner Wissenschaftsnetz“, das im nächsten Abschnitt genauer beschrieben wird, das Lokalisierungssystem „Nyx“ entwickelt. Dieses nutzt zur Ermittlung der Netzkomponente keine klassische Topologieerkennung, sondern Mustererkennung. Realisiert wird diese in Form eines Entscheidungsbaums, der wiederum durch einen maschinellen Lernalgorithmus aus Trainingsdaten erstellt wird. Er klassifiziert die Anschlussports der Netzkomponenten in *Up- bzw. Downlinkports* (Ports zu anderen Switches oder Routern) und *Datenports*, an denen Endgeräte angeschlossen sind.

1.1 Szenario: Münchner Wissenschaftsnetz (MWN)

Das Münchner Wissenschaftsnetz (MWN), das als Szenario für diese Arbeit dient, ist ein sehr großes Kommunikationsnetz, das die Standorte der Münchner Hoch- und Fachhochschulens, viele Studentenwohnheime sowie einige außer-universitäre Einrichtungen umfasst [9]. Betrieben wird das MWN vom Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) und besteht aus einem Backbone-Netz, an dem über Router und Switches die Netze der Einrichtungen an den verschiedenen Standorten angebunden sind. Anhand der nachfolgenden Tabelle 1.1 und der Abbildung 1.1 lässt sich die Komplexität dieses Netzes erahnen.

Kategorie	**Anzahl**
Router (ohne Seriell, DSL und ISDN)	11
Switches	890+
LANs (Subnetze)	1500+
Standorte	60+
Lokale Administratoren	700+
Endgeräte (ohne Wireless)	65.000+
Benutzer	120.000+

Tabelle 1.1: Relevante Fakten zum Münchner Wissenschaftsnetz [7, 9]

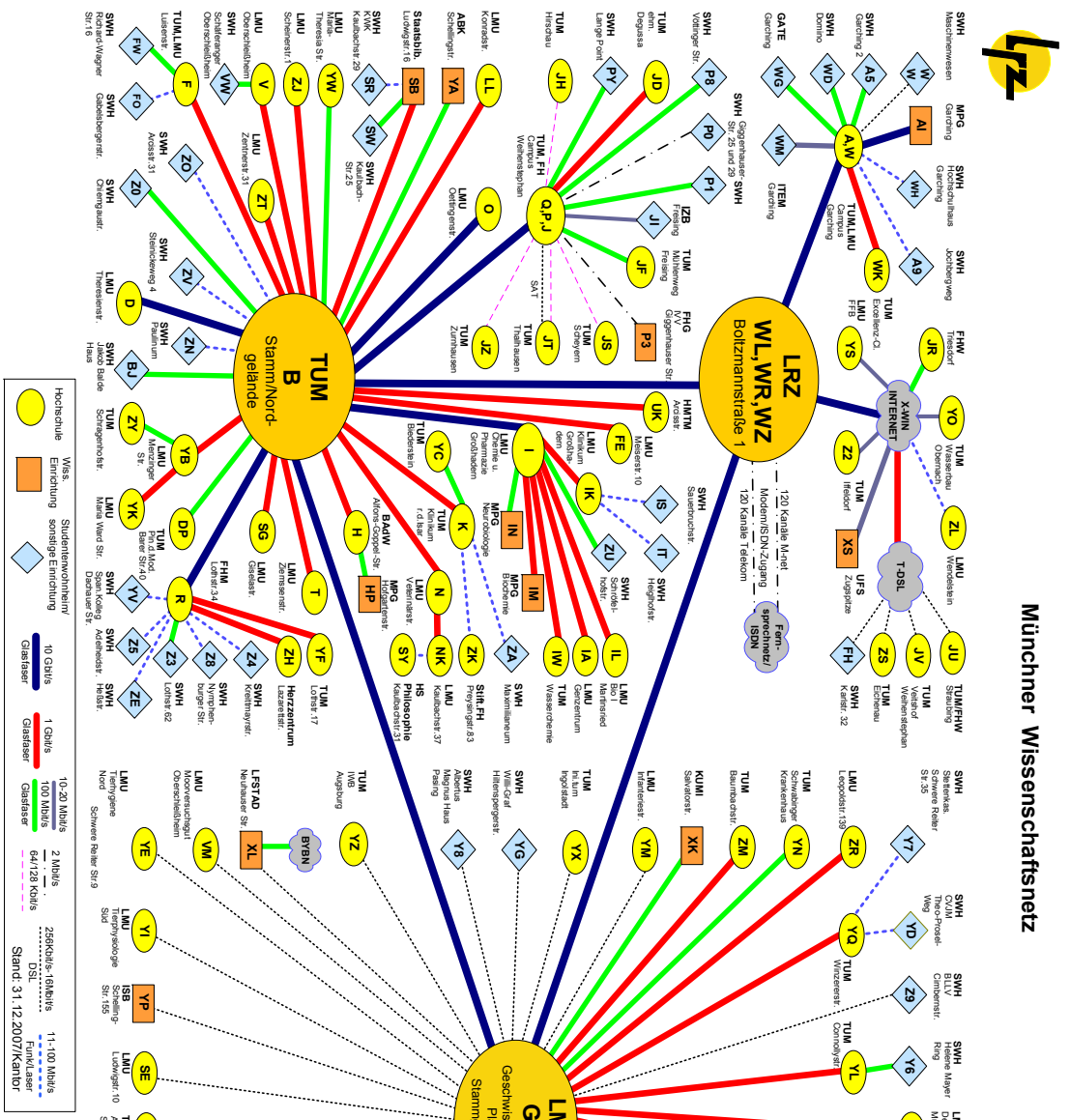


Abbildung 1.1: MWN – Netzstruktur

1.1.1 Anforderungen

Sieht man sich den Daten und Ausmaßen des beschriebenen Szenarios gegenüber, scheint es auf den ersten Blick unmöglich, sich einen kompletten Überblick über die Topologie zu verschaffen und diesen zu jedem Zeitpunkt abrufen zu können. Es muss deshalb zuerst definiert werden, welche Aspekte des Netzes durch ein entsprechendes System festgehalten werden sollen und zu welchem Zweck. Im Rahmen dieser Arbeit geht es nicht darum, die komplette Topologie des gesamten Netzes zu erfassen, sondern lediglich um die Identifikation und Lokalisierung derjenigen Netzkomponente am Rand des Münchner Wissenschaftsnetzes (Edge-Switch), an der ein bestimmtes Endgerät angeschlossen ist. Diese Lokalisierungsinformationen haben in der Praxis vor allem im Zusammenhang mit Abuse-Fällen hohe Relevanz. Dabei zeigen potenziell infizierte bzw. kompromittierte Rechner auffälliges und unerwünschtes Verhalten, wie die folgenden Beispiele verdeutlichen sollen.

Beispiel 1.1 (Versenden von SPAM-Emails) *Ein Rechner, der durch einen Trojaner infiziert worden ist oder zu einem Botnetz gehört, versendet eine sehr hohe Anzahl von Spam-Emails. Es entsteht ein Missbrauch (engl. „abuse“) des Rechners, der (unter Umständen zeitversetzt) festgestellt und gemeldet wird. Zur Behebung muss der Rechner lokalisiert werden. Bemerkung: Der Schaden, der durch Spam im Jahre 2005 in Deutschland nach Studien des Analystenhauses Ferris Research entstanden ist, lag bei rund 4,5 Milliarden US-Dollar und weltweit bei rund 50 Milliarden US-Dollar [15].*

Beispiel 1.2 (Verstöße gegen Sicherheits- oder Benutzerrichtlinien) *Ein Nutzer, der sich innerhalb des MWN aufhält, führt exzessive Port-Scans durch oder handelt wider bekannten Benutzerrichtlinien. Wird diese Zuwiderhandlung festgestellt, kann auch in diesem Fall eine Lokalisierung des betroffenen Endgeräte (und implizit eine Identifikation des Nutzers) notwendig sein, um Gegenmaßnahmen durch lokale Administratoren oder das LRZ selbst einleiten zu können.*

Die beiden Beispiele verdeutlichen, dass in der Praxis die Lokalisierung im Vordergrund steht, weniger die Kenntnis über die komplette Topologie. Die bereits mehrfach angedeutete Lokalisierung erfolgt für die beiden Beispiele (wie für die meisten Fälle in der Praxis) meist ausgehend von der IP-Adresse, die zum MWN gehört und im Folgenden als gegeben angesehen wird. Damit besteht scheinbar bereits an dieser Stelle die Möglichkeit einer Problemlösung, indem man die IP-Adresse am Edge-Router (Bindeglied zwischen dem MWN und dem Internet) sperrt. Allerdings wäre dieses Verfahren erstens nicht dauerhaft (aufgrund von eventueller dynamischer IP-Vergabe) und zweitens unzureichend, da Kommunikation innerhalb des MWN immer noch möglich wäre und im eigenen Netz potenzieller Schaden entstehen könnte.

Die zusätzliche Ermittlung des entsprechenden Endgerätes und des Standorts erlaubt hier eine effektivere und dauerhaftere Lösung des Problems. Mit den Lokalisationsinformationen ist es nämlich möglich, „vor Ort“ Gegenmaßnahmen durchzuführen. Unter der Lokalisation versteht man dabei, wie bereits beschrieben, die Identifikation des Edge-Switches, aber auch des entsprechenden Ports an diesem Switch, d.h. einen möglichst genauen Anschlusspunkt des Endgeräts an das MWN. Sie erfolgt im Wesentlichen über die IP- und MAC-Adressen und sollte (möglichst) eindeutig und aussagekräftig sein.

Aus dieser Problemlösung resultieren weitere Anforderungen an ein Lokalisierungssystem, das trotz der hohen Dynamik im Netz (Aus-/Umbau, Modernisierung, dynamische IP-Vergabe, Standortänderung eines Endgeräts, Hinzukommen/Wegfall von Endgeräten) die Identifikation und Lokalisierung ermöglichen soll. Diese Anforderung bedarf offensichtlich ständiger Überwachung des Netzes, die effektiv nur mit einem hohen Grad an Automatisierung und Parallelisierung erfolgen kann. Als weitere Anforderung sollten außerdem die fehlende Unterstützung für Topologieerkennungsprotokolle und die Heterogenität der Netzkomponenten keine Beeinträchtigungen darstellen. Dabei ist vor allem der zweite Punkt, die Heterogenität, nicht zu vernachlässigen, die durch den Einsatz der folgenden Netzkomponenten zustande kommt:

Router: Cisco Catalyst 6509, 7200, 800, 1700, 1800 Die Router im Backbone sind vom Typ Cisco Catalyst 6509 und unterstützen Ethernet 10/100/1000/10.000 Mbit/s. Ein Cisco 7200 wird zur Anbindung einiger über ISDN-Standleitungen angeschlossener Gebäude in Weihenstephan und einige Geräte der Typenreihen Cisco 800, 1700 und 1800 zur Anbindung einiger über DSL oder seriellen Standleitungen angeschlossener Gebäude betrieben.

Switches (Hersteller: Hewlett Packard): In Gebäudenetzen werden Switches des Typs HP ProCurve eingesetzt. Diese Geräte können bis zu 192 Anschlussports aufnehmen. Der Uplink zum Backbone ist bei manchen zentralen Switches der größeren Standorte über 10 Gigabit-Ethernet, bei den übrigen mit Gigabit-Ethernet realisiert. Die Geräte unterstützen VLAN-Tagging und SNMP-Management. In einigen neueren Gebäuden wird der Typ Procurve 5400 eingesetzt, welcher eine ausreichende Stromversorgung (für Access Points und IP-Telefone) über den Datenanschluss (Power over LAN) unterstützt.

Switches (Hersteller: F5 Networks): Zum Zweck der Lastverteilung und Ausfallsicherheit wichtiger Server werden zwei Layer4/Layer7-Switches (Service Load Balancer) des Typs F5 BigIP 3400 redundant eingesetzt. Dabei handelt es sich korrekter Weise um sogenannte Appliances, die aber auch wesentliche Eigenschaften eines Switches haben. Zurzeit sind WWW (Extern, Intern und Virtuelle), Radius, PAC, HTTP-Proxys und „Webmailer“ angeschlossen.

1.1.2 Ausgewählte vorhandene Lösungsansätze zur Topologieerkennung

Zur Erkennung einer Netztopologie sind bereits verschiedene Ansätze vorhanden, die vielfach Anwendung in der Praxis finden. Ausgewählte Protokolle und Verfahren werden deshalb im Folgenden beschrieben. Für alle Methoden gilt jedoch, dass sie für das Szenario nicht geeignet sind, weil dessen Komplexität aus den bereits beschriebenen Gründen zu hoch ist.

1.1.2.1 Cisco Discovery Protocol (CDP)

Das Cisco Discovery Protocol ist ein proprietäres Protokoll zur Nachbarsuche (Neighbor Discovery), das mit speziellen periodischen Nachrichten (engl. „advertisements“) arbeitet und vom Namensgeber Cisco Systems entwickelt wurde [3]. Es ermöglicht der Netzkomponente, alle CDP-fähigen Nachbarn zu finden, und damit sind Ports genau dann Up- bzw. Downlinkports, wenn daran andere CDP-fähige Netzkomponenten angeschlossen sind.

1.1.2.2 Link Layer Discovery Protocol (LLDP)

Das Link Layer Discovery Protocol ist ein neuer, offener und zugleich herstellerunabhängiger IEEE-Standard [6]. LLDP hat eine ähnliche Funktionalität wie CDP, ist dazu aber inkompatibel. Es gilt aber genau wie bei CDP: Ports sind genau dann Up- bzw. Downlinkports, wenn daran andere LLDP-fähige Geräte angeschlossen sind. Die wesentlichen Vorteile von LLDP im Vergleich zu CDP sind die Herstellerunabhängigkeit und die teilweise bereits vorhandene Unterstützung durch Netzkomponenten im MWN. Die so gewonnenen Informationen können so zwar sinnvoll verwertet werden, es bleiben aber aktuell Lücken durch LLDP-inkompatible Geräte (vor allem durch Cisco-Geräte), sodass LLDP alleine zur Topologieerkennung nicht ausreicht.

1.1.2.3 Manuelle Verfahren

Manuelle Verfahren zur kompletten Topologieerkennung scheitern offensichtlich selbst bei Zuhilfenahme von Software und Datenbanksystemen am hohen Aufwand und der geringen Aktualität. Im Falle des MWN besteht aber eine Abbildung der verwaltbaren Netzkomponenten (Router und Switches des LRZs) in Form einer sogenannten „Netzdokumentation“. Diese kann alleine niemals zur Identifikation und Lokalisierung verwendet werden, allerdings bietet sie möglichweise durch ihre akkuraten Informationen eine Basis für Vergleiche von Topologieerkennungsverfahren.

1.1.3 Nyx: Lokalisierung von Geräten in Echtzeit

Die vorhandenen Ansätze zur Topologieerkennung sind offensichtlich für sich gesehen nicht geeignet, die Anforderungen des beschriebenen Szenarios zu erfüllen. Aus diesem Grund nutzt Nyx diese Informationen zwar sinnvoll, geht aber grundsätzlich einen etwas anderen Weg: Das Lokalisierungssystem nutzt Mustererkennung statt klassischer Topologieerkennung, um Up- bzw. Downlinkports von Datenports zu unterscheiden. Wesentliche Annahme dabei ist, dass beide Portarten charakteristische Eigenschaften aufweisen und sich anhand ihrer Muster klassifizieren (d.h. in Klassen einteilen) lassen. Ist dies der Fall, müssen lediglich die zur Lokalisierung notwendigen Datenports gespeichert werden. Die Up- bzw. Downlinkports hingegen können verworfen werden, da sie keine Auskunft über den Standort eines Endgeräts geben und die Lokalisierung sogar erschweren. Sie erzeugen nämlich überflüssige Daten und verhindern dadurch eindeutige Lokalisierungssaussagen.

Durch den beschriebenen abstrakteren Ansatz reduziert Nyx das Problem auf die Portunterscheidung (Klassifikation) und löst gleichzeitig das Problem der Hardwareheterogenität und der mangelnden Unterstützung von Topologieerkennungsprotokollen.

Damit der Ansatz in der Praxis trotz der hohen Dynamik des Netzes funktioniert, müssen die relevanten Informationen stets in Echtzeit zur Verfügung stehen. Echtzeit bedeutet bei Nyx, dass die Daten im Zeitfenster der Aging-Time der MAC-Forwarding-Tabellen der Netzkomponenten abgefragt werden müssen (vgl. Definition aus [7]). Aus diesem Grund müssen die Informationen über die Anschlussports fortlaufend hoch parallelisiert ermittelt werden.

1.2 Aufgabenstellung und Zielsetzung

Im Rahmen dieser Bachelorarbeit soll die Erkennungsleistung von Nyx erstmals auf mathematisch fundierte Weise quantifiziert werden, da bisher lediglich grobe Schätzungen vorliegen, die im Wesentlichen auf Vermutungen und grundlegenden Analysen basieren. Dazu ist neben der Erarbeitung der theoretischen Grundlagen eine systematische Ermittlung der Fehler- und Erkennungsraten für die Topologie- bzw. Mustererkennung durchzuführen. In diesem Zusammenhang ist insbesondere die Rate von False-Positives und False-Negatives zu bestimmen.

Anschließend soll das System in einem zweiten Schritt – soweit möglich – verbessert werden. Hauptaugenmerk soll dabei auf den Trainingsdaten und dem maschinellen Lernalgorithmus liegen. Diese sollen analysiert, optimiert und evaluiert werden. Im Rahmen der Optimierung sollen durch die Analyse entdeckte Fehler korrigiert und die Erkennungsrate möglichst gesteigert werden.

Ziel dieser Bachelorarbeit ist also, die Erkennungsleistung statistisch exakt zu ermitteln und diese im Rahmen der Möglichkeiten zu verbessern. Es wird versucht, die Analyse zu automatisieren, um trotz der großen Anzahl an Netzkomponenten und Endgeräten eine realistische Aussage über das gesamte Netz treffen zu können. Dies wird manuell sicher kaum möglich sein, trotzdem wird auch diese Möglichkeit untersucht. Potentielle Verbesserungsmöglichkeiten sind an dieser Stelle noch vollkommen unbekannt, sodass hier keine Aussage über eine konkrete Zielsetzung gemacht werden kann. Im Allgemeinen wird natürlich versucht, möglichst alle entdeckten Fehler zu korrigieren und die Leistung des Systems zu optimieren.

1.3 Aufbau der Arbeit

Nach dieser Einleitung folgt in Kapitel 2 eine Erklärung des Lokalisierungssystems Nyx. Es werden grundlegende Konzepte für die szenariobasierte Lokalisierung von Endgeräten erklärt, insbesondere auch die Theorie maschineller Lernverfahren. Besonderer Schwerpunkt liegt dabei auf dem Algorithmus J4.8 und dessen Klassifikationsmodell, dem Entscheidungsbaum.

Das dritte Kapitel beschreibt darauf aufbauend verschiedene Möglichkeiten der Analyse dieses Lokalisierungssystems. Es werden manuelle, semi-automatisierte und vollständig-automatisierte Verfahren vorgestellt. Wesentlicher Schwerpunkt dieses Kapitels ist ein automatisches Analyse-Skript, das im Rahmen dieser Arbeit entwickelt wurde. Abschließend wird dieses zur Ermittlung des Ist-Zustands verwendet, der als Vergleich für die darauffolgenden Verbesserungen dienen soll.

Die Verbesserungen selbst werden in Kapitel 4 beschrieben. Sie umfassen alle Optimierungen der Trainingsdaten sowie die Veränderungen am Lernalgorithmus. Bewertet werden sie am Ende des Kapitels ebenfalls mit dem entwickelten Analyse-Skript. Abschließend werden die Ergebnisse dann in einem Fazit zusammengefasst und ein Ausblick auf deren zukünftige Bedeutung gegeben.

2 Nyx: ein Echtzeit-Lokalisationssystem

Wie bereits im vorherigen Kapitel beschrieben, ist Nyx ein Lokalisierungssystem, das vom LRZ entwickelt worden ist. Die wesentlichen Komponenten von Nyx dienen zur Topologie- bzw. Mustererkennung und sind konzeptionell auf hohe Parallelität zur Erfüllung der Echtzeitanforderungen ausgelegt. Als Eingabe erhält Nyx in den meisten Fälle eine IP-Adresse, mit der eventuell bereits eine äußerst grobe Eingrenzung bei der Lokalisierung anhand der systematisch vergebenen Subnetze erfolgen kann. Diese ist aber für die definierten Anforderungen nicht ausreichend. Nyx verwendet deshalb zur (genaueren) Lokalisierung die gerätespezifische MAC-Adresse.

2.1 Die Lokalisierung durch Portklassifikation

Ausgehend von der gegebenen IP-Adresse wird basierend auf dem Address Resolution Protocol (ARP) durch den Router die entsprechende MAC-Adresse ermittelt. Nyx versucht dann für diese MAC-Adresse durch zuvor gesammelte Lokalisationsinformationen, den entsprechenden Edge-Switch ausfindig zu machen, an dem das Endgerät angeschlossen ist. Diese Lokalisation ist nicht trivial, da die betrachtete MAC-Adresse nicht nur am Edge-Switch, sondern auf dem gesamten Kommunikationspfad zum Router (mindestens) an jedem Downlinkport einer Netzkomponente, die zur direkten Kommunikationskette gehört, auftritt (siehe dazu Abbildung 2.1 in Anlehnung an [7]).

Aus diesem Grund ist zwingend eine Filterung notwendig, damit die Datenbank nur möglichst wenige, aber alle nötigen Lokalisationsinformationen speichern muss und die geforderte Funktionalität überhaupt gewährleistet ist. Dazu muss der Edge-Switch von allen anderen inneren Switches (in Bezug auf die aktuell betrachtete MAC-Adresse) unterschieden werden.

Graphentheoretisch betrachtet bedeutet das, die inneren Knoten (Switches) einschließlich der Wurzel, die zum Pfad von der Wurzel zum Blatt (und umgekehrt) gehören, vom Blatt (Endgerät) zu unterscheiden. Diese Unterscheidung entspricht außerdem der Einteilung der Verbindungen in „Inter-Switch-Verbindungen“ und „Endgeräteverbindungen“ (siehe dazu Abbildung 3.2 auf Seite 27). Bei Inter-Switch-Verbindungen sind beide Netzkomponenten Switches (Router seien vernachlässigt) und die beteiligten Ports sind Up- bzw. Downlinkports. Bei Endgeräte-Verbindungen handelt es sich bei dem Port um einen Datenport, der von einem Switch zu einem Endgerät geht.

Unter der Annahme einer entsprechenden Unterscheidung können die Daten zu den inneren Knoten gelöscht werden und lediglich die Daten zum Edge-Switch persistent in die Datenbank eingetragen werden. Offensichtlich sind bei diesem Vorgehen also nur genügend Informationen zur Unterscheidung nötig, aber keine komplette strikte Topologieerkennung, wie man vielleicht vermuten würde. Nyx führt demnach eigentlich keine Topologieerkennung im klassischen Sinne durch, sondern eine Klassifikation durch Mustererkennung, die auf maschinellem Lernen und vorher zur Verfügung gestellten (im Wesentlichen statischen) Trainingsdaten basiert. Dieses Verfahren wird auch „supervised learning“ genannt und im folgenden Abschnitt genauer beschrieben. Es hat den Vorteil, dass weiche Kriterien (wie die Anzahl der MAC-Adressen) verwendet werden können, statt diese (unveränderlich) im Programm festzulegen. Die Trai-

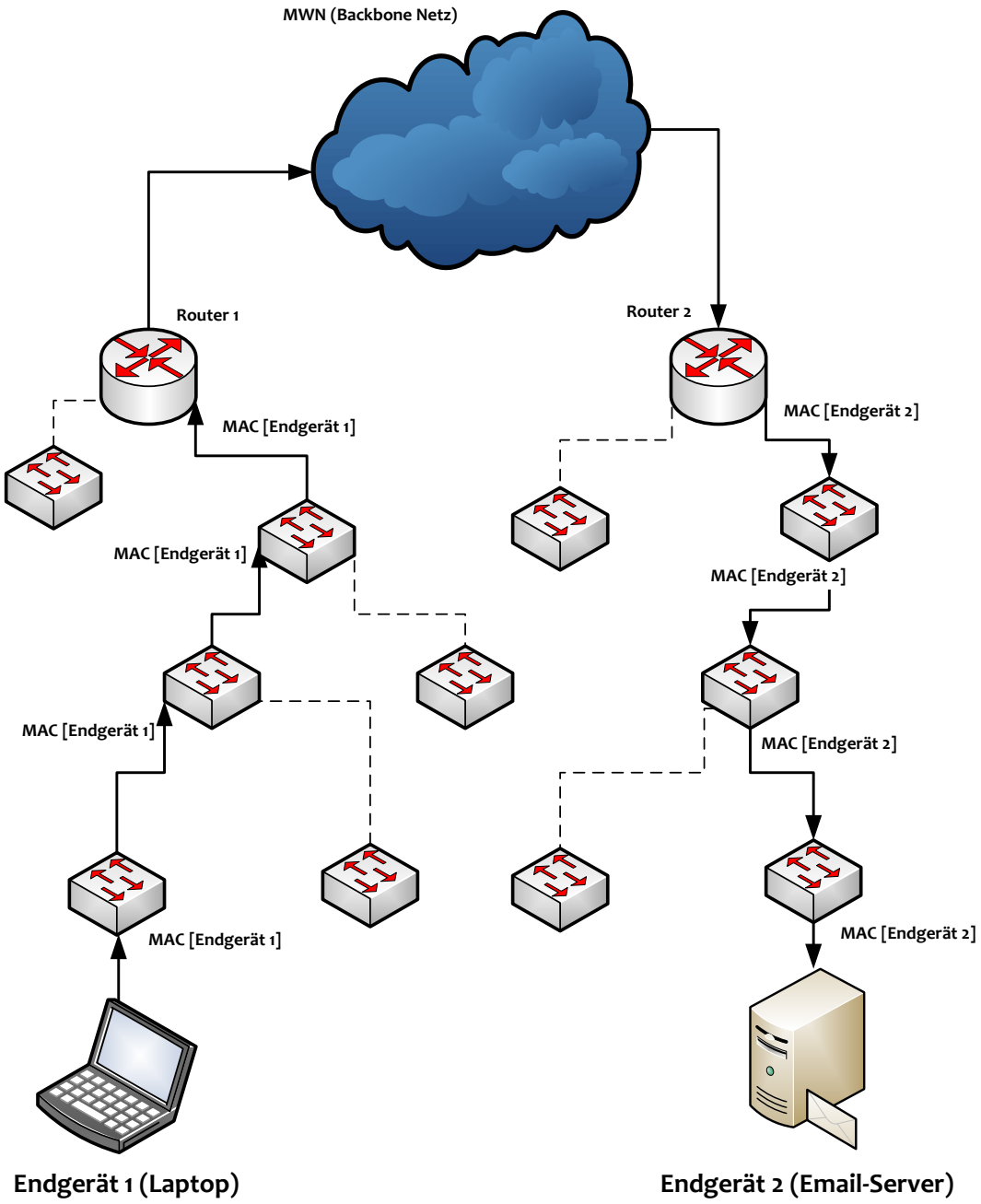


Abbildung 2.1: Topologie-Beispiel: Die MAC-Adressen der beiden Endgeräte sind an jedem Switch bis zum Router (am Downlinkport) sichtbar.

ningsdaten sind statistische Daten zu ausgewählten Ports, die in Abschnitt 2.2.3 genauer beschrieben werden.

Das Ergebnis des maschinellen Lernalgorithmus ist eine Klassifizierung aller Ports in die Klassen *Uplinkport* bzw. *Downlinkport* und *Datenport*. Da die Datenflussrichtung aber offensichtlich keine Rolle spielt, weil lediglich eine Unterscheidung von Inter-Switch- und Endgeräteverbindung erfolgen muss und keine klassische Topologieerkennung erfolgt, kann die Klassifizierung in Up- bzw. Downlinkport zusammengefasst werden. Diese Klassifikationsdaten bilden einen wesentlichen Teil der Lokalisationsdaten, die zusätzlich weitere Informationen beispielsweise zu den Netzkomponenten und Ports enthalten.

2.2 Maschinelles Lernen bei Nyx

Unter „Lernen“ versteht man im Allgemeinen den zielgerichteten (= intentionales Lernen) und den beiläufigen (= inzidentelles und implizites Lernen), individuellen oder kollektiven Gewinn von (grundlegendem) Wissen, Kenntnissen, Fertigkeiten, Fähigkeiten und/oder nützlichen Verhaltensmustern. Maschinelles Lernen im Speziellen ist vornehmlich der Sammelbegriff für die „künstliche“ Generierung von Wissen: In der Lernphase extrahiert ein künstliches System aus Lerndaten allgemeingültiges Wissen, d.h. es versucht für die Daten grundlegende Gesetzmäßigkeiten zu erkennen. Das Ermitteln dieser Gesetzmäßigkeiten entspricht dem eigentlichen Lernen, da im Zusammenhang mit maschinellem Lernen kein konkretes Wissen, wie es Datenbanken vorhalten, gefordert ist. Vielmehr sollte es dem System nach Abschluss der Lernphase prinzipiell möglich sein, auch unbekannte Daten zu beurteilen. Dies ist besonders im Bereich der „Klassifikation“ auf Basis von gelernten Mustern und damit für die Software Nyx entscheidend.

Maschinelles Lernen als Forschungsgebiet, das sich mit der computergestützten Modellierung und Realisierung von Lernphänomenen beschäftigt [17], unterscheidet im Bereich der Klassifikation im Wesentlichen symbolische Lernverfahren, Reinforcement Learning, konnektionistische Lernverfahren (Neuronale Netze), genetische (evolutionäre) Algorithmen und Wissensentdeckung (Data-Mining). Auf die verschiedenen Verfahren wird an dieser Stelle nicht weiter eingegangen, da die in Nyx verwendeten Algorithmen lediglich symbolische Lernverfahren nutzen. Für die Zwecke dieser Arbeit genügt deshalb die allgemeine Unterscheidung bezüglich Art und Mächtigkeit der Wissensrepräsentation in symbolische Systeme, in denen das Wissen – sowohl die Beispiele als auch die induzierten Regeln – explizit repräsentiert ist, und subsymbolische Systeme (wie neuronale Netze), denen zwar ein berechenbares Verhalten „antrainiert“ wird, die jedoch keinen Einblick in die erlernten Lösungswege erlauben; hier ist Wissen implizit repräsentiert. Vertreter der ersten Gruppe sind ID3 und seine Nachfolger C4.5 und J4.8 [13].

2.2.1 Überwachtes Lernen: „Supervised Learning“

Die praktische Umsetzung des maschinellen Lernens erfolgt meist mittels Lernalgorithmen. Unter einem Algorithmus versteht man im Allgemeinen die genaue Beschreibung eines Verfahrens zur Lösung eines Problems. Im Bereich des maschinellen Lernens existieren im Wesentlichen drei Klassen zur Einordnung der Algorithmen, wobei vor allem die erste Klasse für diese Arbeit relevant ist.

Klassen des maschinellen Lernens:

Überwachtes Lernen (engl. „supervised learning“): Der eingesetzte Algorithmus lernt ein Modell bzw. eine Funktion aus vorgegebenen Lerndaten (Paaren von Ein- und Ausgaben), die auch als Trainingsdaten bezeichnet werden und bereits korrekte Ausgabewerte zu einer Eingabe beinhalten. Ein Teilgebiet dieses überwachten Lernens ist die automatische Klassifizierung, die bei Nyx eine wesentliche Rolle einnimmt. Diese Klasse wird im Anschluss an diese Übersicht genauer beschrieben.

Nicht-überwachtes Lernen (engl. „unsupervised learning“): Der Algorithmus generiert für eine gegebene Menge von Eingaben (ohne entsprechende Ausgaben) ein Modell, das allein die Eingaben beschreibt und dadurch Prognosen ermöglicht.

Zu dieser Klasse gehören Clustering-Verfahren, die Eingaben anhand von charakteristischen Mustern bzw. Merkmalen voneinander unterscheiden und in Klassen einteilen. Ein wichtiger Vertreter dieser Verfahren ist der EM-Algorithmus, der die Parameter eines Modells iterativ so festlegt, dass die gesehenen Daten optimal erklärt werden. Eine Anwendung des EM-Algorithmus findet sich beispielsweise in den Hidden Markov Models (HMMs).

Bekräftigungslernen (engl. „reinforcement learning“): Der Algorithmus lernt durch „Belohnung“ und „Bestrafung“ für gewisse Aktionen ein Verhaltensmuster, um den Nutzen des Agenten (d. h. des Systems, zu dem die Lernkomponente gehört) bei potenziell auftretenden Situationen zu maximieren. Man nennt dieses Lernen auch „Lernen aus Versuch und Irrtum“ oder „schwach überwachtes Lernen“.

Das bereits angesprochene Supervised Learning wird bei Nyx für die automatische Klassifikation von Up- bzw. Downlinkports und Datenports verwendet. Nyx lernt aus vorgegebenen Beispielen (Trainingsdaten, engl. „seed data“) eine Klassifikationshypothese: Ausgehend von einer Folge von Lernbeispielen (**Eingabedaten**)

$$e_1, e_2, ...e_n$$

erzeugt der verwendete Lernalgorithmus im Allgemeinen eine **Hypothese** h_n, die (in der Theorie) alle Trainingsdaten $e_1, e_2, ...e_n$ korrekt beschreibt. Ist die Hypothese im Sinne des zu lernenden Konzepts noch nicht endgültig (Endhypothese), dann wird sie unter Verwendung weiterer Datensätze aus der Eingabe e_{n+1} zu einer neuen Hypothese h_{n+1} modifiziert und erweitert (in Anlehnung an [17]).

Nach Abschluss der Lernphase wird davon ausgegangen, dass das zu lernende Konzept korrekt in Form der Endhypothese vorliegt. Das kann anhand von Validierungsmechanismen überprüft werden, die im weiteren Verlauf dieser Arbeit noch genauer beschrieben werden.

2.2.2 Waikato Environment for Knowledge Analysis (Weka)

Weka (Waikato Environment for Knowledge Analysis, [20]) ist eine Open-Source-Sammlung von maschinellen Lernalgorithmen zum Data-Mining, die von der Universität Waikato entwickelt wurde und unter GNU General Public License frei verfügbar ist. Die unterstützten Algorithmen können entweder direkt auf ein Datenset angewendet werden oder in eigene JAVA-Programme eingebunden werden, wie es bei Nyx der Fall ist. Weka bietet dabei Möglichkeiten zum sogenannten Pre-Processing, zur Klassifikation, Visualisierung und weitere Funktionen, die aber für Nyx keine weitere Bedeutung haben.

2.2.3 Relevante Eingabedaten zur Klassifikation bei Nyx

Zur Klassifikation verwendet Nyx die Verkehrsdaten der Netzkomponenten. Diese Verkehrsinformationen werden mittels Simple Network Management Protocol (SNMP) abgefragt. SNMP ist ein Netzwerkprotokoll, das von der Internet Engineering Task Force (IETF) entwickelt wurde [5], um verschiedenste Netzwerkelemente (z. B. Router, Switches, Server, Drucker, Computer usw.) standardisiert von einer zentralen Station aus überwachen und steuern zu können. Zusätzlich werden weitere Informationen bei der Klassifizierung berücksichtigt, die zum Teil aus Topologieerkennungsprotokollen resultieren und die Klassifikation untermauern sollen.

Klassifikationskriterien aus der Implementierung von Nyx:

macs_avg: Dem Kriterium „Anzahl der durchschnittlichen MAC-Adressen“ (macs_avg) liegt die Annahme zu Grunde, dass eine hohe Zahl von Adressen normalerweise nur an einem Up- bzw. Downlinkport vorkommt, weil ein Endgerät selten mehrere (im Netz aktive) MAC-Adressen hat. Prinzipiell kann man sich aber schon vorstellen, dass es für diese Annahme auch Gegenbeispiele gibt (Server, virtualisierte Komponenten, Geräte hinter sogenannten nicht SNMP-fähigen Mini-Switches, ...). Dieses Kriterium wird deshalb nicht fest in das Programm Nyx implementiert, sondern als Lernkriterium behandelt.

traffic_percent_avg Ein weiteres Kriterium stellt der durchschnittliche Anteil des betrachteten Ports am Datenverkehr dar. Um diesen zu ermitteln wird im Wesentlichen der gesamte Datenverkehr prozentual anhand der verbundenen Geräte auf die einzelnen Ports verteilt. Die Aussagekraft dieses Kriteriums wird ebenfalls in Abschnitt 3.5 kritisch diskutiert.

component_mac_discovered: Das zweite Kriterium für die Klassifikation eines Ports ist das zweiwertige Attribut „component_mac_discovered“. Es gibt an, ob an dem aktuell betrachteten Port eine bekannte MAC-Adresse einer Netzkomponente erkannt worden ist. Wenn dies der Fall ist, kann man davon ausgehen, dass es sich um einen Up- bzw. Downlinkport handelt.

component_discovered: Dieses Kriterium ist ebenfalls ein zweiwertiges Attribut eines Ports, das angibt, ob eine angeschlossene LLDP-fähige Netzkomponente erkannt worden ist. Hier wird also das Topologieprotokoll LLDP konstruktiv in die Klassifikationsentscheidung eingebunden. Wurde eine Netzkomponente erkannt, handelt es sich bei dem aktuell betrachteten Port sehr wahrscheinlich um einen Up- bzw. Downlinkport.

tagged: Das zweiwertige Kriterium „tagged“ hält fest, ob der Port im Rahmen des Einsatzes von VLANs (Virtual Local Area Network) markiert wurde. Ist dies der Fall, tragen alle Netzpakete eine zusätzliche Markierung, die sie einem bestimmten VLAN zuordnet. Dieses sogenannte „VLAN tagging“ kommt hauptsächlich an Up- bzw. Downlinkports zum Einsatz. Es ist aber auch möglich, dass Server über VLAN-tagged Ports verfügen.

custom_criteria1: Das letzte Kriterium erlaubt das manuelle Eintragen einer verbundenen Netzkomponente. Dieses Kriterium ist relativ sicher, wird aber nur sehr selten verwendet, um eine Inter-Switch-Verbindung zu beschreiben.

2.2.4 Verwendeter Klassifizierungsalgorithmus: J4.8

In Nyx kommt der Algorithmus J4.8 als Klassifizierer zum Einsatz. J4.8 ist eine jüngere und leicht verbesserte Version von C4.5 Revision 8, der wiederum eine Weiterentwicklung des Algorithmus ID3 ist und die letzte frei verfügbare Version dieser Algorithmenfamilie darstellt [20, 13]. J4.8 als Nachfolger ist in JAVA implementiert, was auch das „J“ in der Bezeichnung erklärt. Der Algorithmus hat dadurch den wesentlichen Vorteil der Plattformunabhängigkeit, die jedoch Performanzeinbußen gegenüber der Originalversion in C nach sich zieht.

Dem Algorithmus J4.8 liegt – genau wie dem Algorithmus C4.5 – ein Entscheidungsbaum als Klassifikationsmodell zugrunde. Diese Art von Modell gehört zu den symbolischen Lernverfahren, weil das Wissen – sowohl die Beispiele als auch die induzierten Regeln – explizit repräsentiert sind. Damit ist es möglich, das erlernte Modell anhand von konkreten Beispielfällen zu testen. Das ist bei subsymbolischen Systemen, in denen das Wissen nur implizit repräsentiert wird, nur schwer oder gar nicht möglich. Es hat also Vorteile, dass bei Nyx gerade ein Algorithmus aus der Klasse der symbolischen Lernverfahren verwendet wird, obwohl auch andere Algorithmen zur Auswahl stehen und anwendbar wären.

Ein Klassifikationsmodell ist eine Verbindung zwischen Attributen (Eigenschaften) und Klassen, das beispielsweise durch ein einfaches Flussdiagramm beschrieben werden kann. Es wird im Fall von J4.8 aus einer gewissen Anzahl von Eingabedaten (hier: Klassifikationen) induktiv, d.h. durch Untersuchung und Generalisierung vom Speziellen zum Allgemeinen (engl. „bottom-up“), konstruiert. Dazu sind offensichtlich genügend Trainingsdaten nötig, damit eine solche Generalisierung mit zufriedenstellenden Ergebnissen erfolgen kann.

Die Lernbeispiele (engl. „case“ [Singular]) müssen in einer festen Form und Ordnung und in Textform angegeben werden können, wobei jede Zeile einen Fall beinhaltet (engl. „flat file“), dessen Felder bzw. Attribute (meist) durch Feldbegrenzer (z.B. Kommata) separiert sind. Die Attribute selbst können entweder diskret („nominal“) oder numerisch („numeric“) vorliegen und dieser Typ darf sich nicht von Fall zu Fall unterscheiden. Die Typen sind deshalb meist am Anfang der Eingabedaten – ähnlich wie bei Programmiersprachen – festzulegen, sodass eine Überprüfung durch den Algorithmus möglich ist.

Bei überwachtem Lernen müssen zusätzlich zu den Attributen die Klassen des Klassifikationsmodell ex ante, d.h. bei Erstellung der Eingabedaten, definiert werden. Dies wäre bei unüberwachtem Lernen nicht der Fall, da dort die Klassen ex post durch Analyse generiert werden. Da der eingesetzte Algorithmus J4.8 aber zur Klasse der überwachten Lernverfahren gehört, ist diese Definition nötig. Die Klassendefinition muss in diesem Fall außerdem diskret sein, d.h. sie muss disjunkt (ohne Überschneidung) sein, damit eine eindeutige Zuordnung der zu klassifizierenden Daten erfolgen kann.

2.2.4.1 Grundlagen zum Klassifikationsmodell „Entscheidungsbaum“

Dem Algorithmus J4.8 liegt als Klassifikationsmodell ein Entscheidungsbaum (engl. „decision tree“) zugrunde. Ein Entscheidungsbaum ist eine (Daten-)Struktur und besteht entweder aus

- einem **Blatt**, das eine Entscheidung bzw. Klasse enthält, oder
- einem (kleineren) **Baum**, dessen Wurzel ein Entscheidungsknoten ist und einen Vergleich bzw. Test auf Basis eines einzelnen Attributs enthält.

(in Anlehnung an [13, Seite 5 (1.1.1 Decision trees)])

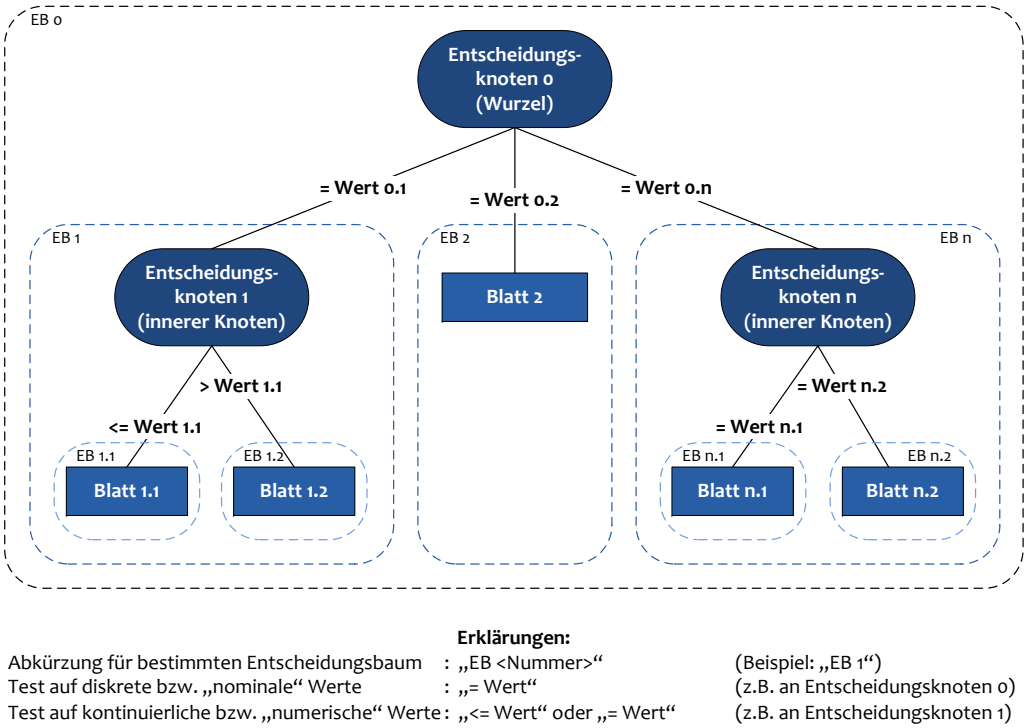


Abbildung 2.2: Prinzipieller Aufbau eines Entscheidungsbaums

Die Klassifikation eines unbekannten Datensatzes erfolgt durch Pfadtraversierung beginnend mit der Wurzel. Der Pfad entsteht dabei dynamisch in Abhängigkeit von den Ergebnissen der Vergleiche bzw. Tests in den Entscheidungsknoten. Der jeweils aktuelle Test sei dabei im Folgenden (ohne weitere formale Unterscheidung von vorherigen oder nachfolgenden Tests) mit X bezeichnet. Beim verwendeten Klassifikationsalgorithmus J4.8 können diese Tests diskreter oder kontinuierlicher Natur sein. Im ersten Fall wird auf Übereinstimmung mit einem diskreten („nominalen“) Wert geprüft, im zweiten werden meist Vergleiche für gewisse („numerische“) Wertebereiche durchgeführt. Beide Fällen sind in Abbildung 2.2 in Form der Entscheidungsknoten 0 und 1 dargestellt.

Abbruchbedingung für dieses rekursive Vorgehen, in dem stets der Test X durchgeführt und für den neuen Teilbaum das Verfahren wiederholt wird, ist das Erreichen eines Blattknotens. Diese Abbruchbedingung tritt bei einer Baumstruktur immer nach endlich vielen Schritten ein. Der Beweis lässt sich leicht anhand einer einfachen Induktion über die oben beschriebene Baumstruktur und der endlichen Anzahl an Knoten führen. Das Resultat der Klassifikation ist folglich immer die im Blattknoten enthaltene Klasse.

Die Konstruktion eines solchen Entscheidungsbaums ist neben der eigentlichen Klassifikation die wesentliche Aufgabe eines Klassifikationsalgorithmus. Der grundlegende Ansatz für die Erstellung des Modells ist bekannt als „divide-and-conquer"-Verfahren mittels eines Greedy-Algorithmus ohne sogenanntes „Backtracking" (genauere Ausführungen dazu in [13, 18]).

Sei T die Menge an Trainingsdaten. Die Klassen (engl. „classes") seien $\{C_1, C_2, \ldots, C_n\}$. Für die Konstruktion eines Baums gibt es damit drei Fälle, die betrachtet werden müssen:

1. **Die Menge der (aktuell betrachteten) Trainingsdaten T ist nicht leer und alle Trainingsfälle aus T gehören zu einer Klasse C_k (wobei $k \in \{1, 2, \ldots, n\}$):** Der Entscheidungsbaum für T besteht aus einem Blatt. Alle Trainingsdaten werden durch dieses Blatt repräsentiert.

2. **Die Menge der (aktuell betrachteten) Trainingsdaten T ist leer:** Der Entscheidungsbaum ist wieder ein einzelnes Blatt. In diesem Fall ist die Ausprägung der Klasse im Blatt aber nicht durch die Trainingsdaten, sondern durch zusätzliche Informationen außerhalb der Menge T bestimmt. Der Algorithmus C4.5 wählt diese, indem diejenige Klasse mit den meisten Elementen im Elternknoten bestimmt und übernommen wird.

3. **Die Menge Trainingsdaten T ist nicht leer und die Trainingsfälle gehören zu endlich vielen, verschiedenen Klassen C_i mit $i \in \mathbb{N}, 1 \leq i \leq n$ und $i_1 \neq i_2$:** In diesem Fall wird T in die Teilmengen $T_1, T_2 \ldots, T_n$ zerlegt. Dazu wird ein Vergleich bzw. Test auf Basis eines einzelnen Attributs durchgeführt, dessen Ergebnisse (engl. „outcomes") diskret und allgemein mit $\{O_1, O_2, \ldots, O_n\}$ bezeichnet seien. Die Auswahl des Testattributs erfolgt hierbei, indem das Attribut mit größtem Informationszugewinn – ermittelt anhand des sogenannten Gain-Ratio-Kriteriums (engl. „gain ratio criterion", siehe Abschnitt 2.2.4.2) – als Testkriterium gewählt wird. Alle Datensätze aus T mit dem Testergebnis O_i gehören dann nach Test und Partitionierung in die Teilmenge T_i. Der Entscheidungsbaum erhält dadurch einen neuen Entscheidungsknoten und einen Ast für jedes Ergebnis O_i. Dieses Verfahren wird anschließend rekursiv auf alle Teilmengen T_i angewendet bis einer der beiden vorherigen Fälle eintritt und die Rekursion abbricht. Der Beweis, dass diese Rekursion tatsächlich immer abbricht, ist leicht mittels Induktion mit Fallunterscheidung über die Elemente in der Menge T zu führen.

(in Anlehnung an [13, Seite 17 (2.1 Divide and conquer)])

Am Ende des Verfahrens liegt dann der komplette Entscheidungsbaum vor und kann als Modell zur Klassifikation unbekannter Datensätze verwendet werden.

2.2.4.2 Das Gain- und Gain-Ratio-Kriterium

Der im vorherigen Abschnitt beschriebene Greedy-Algorithmus zur Konstruktion von Entscheidungsbäumen muss im Wesentlichen entscheiden, welcher Test in den inneren Entscheidungsknoten auf welcher Hierarchieebene durchgeführt wird, um ein möglichst vollständiges und korrektes Modell zu erzeugen. Dazu wird beim J4.8 das sogenannte Gain-Ratio-Kriterium verwendet, das nachfolgend basierend auf dem Gain-Kriterium eingeführt wird. Um dabei eine einheitliche Definition zu gewährleisten, gelten die Ausführungen zur Trainingsmenge T, deren Teilmengen T_i, zu den Tests X und dessen Ergebnissen O_i aus dem vorherigen Abschnitt

weiterhin. Zusätzlich gelte, dass zur Beurteilung eines Testkriteriums nur die Verteilung der Klassen innerhalb der Trainingsmenge T und deren Teilmengen T_i zur Verfügung steht, da es sich bei dem beschriebenen Verfahren um einen Greedy-Algorithmus handelt. Aufgrunddessen ist offensichtlich die (absolute) Häufigkeit der Vorkommen einer bestimmten Klasse wichtig. Diese Zahl soll im Allgemeinen mit $freq(C_i, S)$ bezeichnet werden und die Anzahl der Fälle (Datensätze) in S angeben, die zur Klasse C_i gehören. Außerdem gilt die Standardnotation für die Mächtigkeit einer Menge in der Form $|S|$.

Des Weiteren gilt eine wesentliche Erkenntnis aus der Informationstheorie: „Die durch eine Nachricht übermittelte Information hängt von deren (Auftritts-)Wahrscheinlichkeit ab und kann in der Einheit *bits* durch den negativen Logarithmus dualis (Logarithmus zur Basis 2) dieser Wahrscheinlichkeit angegeben werden“ (vgl. [13, 16]).

Beispiel 2.1 (Berechnung der Anzahl an bits) *Existieren insgesamt vier gleich wahrscheinliche Nachrichten, beträgt die Information je übertragener Nachricht*

$$-\log_2\left(\frac{1}{4}\right) = \log_2(4) = 2 \quad bits$$

denn die Nachrichten können offensichtlich durch 00, 01, 10 und 11 codiert werden.

Im ersten Schritt zur Bestimmung des Gain-Wertes gilt folglich im Allgemeinen für die relative Häufigkeit einer Klasse C_i aus S

$$\frac{freq(C_i, S)}{|S|}$$

und damit kann die übertragene Information mittels

$$-\log_2\left(\frac{freq(C_i, S)}{|S|}\right) \quad bits$$

berechnet werden.

Zur Ermittlung der Information, zu welcher Klasse eine bestimmte, informationstheoretische Nachricht gehört, wird über die Klassen im Verhältnis zu deren Auftrittshäufigkeit in S summiert:

$$\text{info}(S) = -\sum_{i=1}^{k} \frac{freq(C_i, S)}{|S|} \times \log_2\left(\frac{freq(C_i, S)}{|S|}\right) \quad \text{bits (= Entropie von S)}$$

Angewendet auf die Trainingsmenge T des Klassifikationsverfahrens gibt info(T) damit also den durchschnittlichen Betrag an Informationen an, der zwingend nötig ist, um die Klasse eines Datensatzes zu identifizieren. Dieser Wert ist entscheidend zur Berechnung des Informationszugewinns, die nachfolgend beschrieben wird, und wird als Entropie bezeichnet.

Im zweiten Schritt zur Berechnung des Gain-Wertes wird die Information nach einer Partitionierung der Trainingsmenge T in n Teilmengen T_i bestimmt. Es gilt wieder wie im vorhe-

rigen Abschnitt, dass alle Datensätze in T_i die gleichen Ergebnissen O_i in Bezug auf den Test X haben. Daraus folgt:

$$\text{info}_X(T) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \text{info}(T_i) \quad \text{bits}$$

Anhand dieser beiden berechneten Werte ergibt sich dann der **Gain**-Wert als

$$\text{gain}(X) = \text{info}(T) - \text{info}_X(T)$$

und misst den dazugewonnenen (engl. „gained“) Informationsgehalt durch die Partitionierung. Das Gain-Kriterium wählt dementsprechend immer denjenigen Test aus der Menge aller möglichen Tests mit dem größten Informationszuwachs (engl. „information gain“).

Das für den J4.8 relevantere Gain-Ratio-Kriterium ist eine „mathematische Weiterentwicklung“ dieses Gain-Kriteriums. Dieses weist nämlich einen bedeutenden systematischen Fehler („Bias“) in Bezug auf Tests mit einer hohen Anzahl von Ergebnissen auf [13, Seite 23 (2.2.2 Gain ration criterion)].

Beispiel 2.2 (Systematischer Fehler) *Der Bias lässt sich sehr leicht nachvollziehen, wenn man sich eine Trainingsmenge vorstellt, die anhand eines einzigartigen Kriteriums (z. B. eine ID) in Teilmengen zerlegt werden kann. Die Teilmengen T_i enthalten in diesem Fall nur einen Datensatz. Der Informationszugewinn ist damit zwar maximal, weil $info_X(T) = 0$ gilt, aber eine solche Partitionierung ist offensichtlich für eine Klassifikation weiterer unbekannter Daten vollkommen sinnlos.*

Man normiert daher das Gain-Kriterium und erhält das **Gain-Ratio-Kriterium** über

$$\text{split\_info}(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right),$$

indem man die Gain-Ratio folgendermaßen berechnet

$$\text{gain\_ratio}(X) = \frac{\text{gain}(X)}{\text{split\_info}(X)}.$$

Dieser Zahlenwert beschreibt also nur den Anteil **nützlicher** Information, die durch die Partitionierung erzeugt wurde. Angemerkt werden muss allerdings, dass annähernd triviale Partitionierungen eine niedrige Partitionsinformation (split_info) haben und damit die gain ratio instabil ist. Deshalb wählt das Gain-Kriterium immer Tests zur Maximierung des split_info-Wert (gemäß der Bedingung, dass der Informationszugewinn groß sein soll).

Beispiel 2.3 (Fortsetzung: Systematischen Fehler) *Beim Gain-Ratio-Kriterium wird die Information einer ID als Kriterium offensichtlich nur als gering eingestuft, weil der Dividend (information gain) höchstens $\log_2(k)$ und der Divisor $\log_2(|T|)$ ist, wobei k die Anzahl der Klassen und $|T| \gg k$. Jeder Datensatz hat nämlich ein einzigartiges Ergebnis in Bezug auf den Test X, also ist der Gain-Ratio-Wert klein.*

Das Gain-Ratio-Kriterium ist damit robust, hat außerdem Vorteile bei binären Tests, auf die an dieser Stelle nicht weiter eingegangen wird, und ist insgesamt besser für Klassifikationsaufgaben geeignet als das Gain-Kriterium, wie man anhand des Beispiels sieht [14]. Das Gain-Ratio-Kriterium wird deshalb auch bei der Analyse von Nyx, die im nächsten Kapitel beschrieben wird, eine wichtige Rolle spielen.

2.2.4.3 Anwendung des Algorithmus auf ein Standardbeispiel

Zur Verdeutlichung des Algorithmus zur Konstruktion eines Entscheidungsbaums bedient man sich im Allgemeinen gewissen Standardbeispielen aus dem Bereich des Data Mining. Ein sehr bekanntes Beispiel ist ein Trainingsset mit Datensätzen zur Klassifikation von Wetterdaten im Zusammenhang mit der Frage, ob man bei einer gegebenen Wetterlage Golf spielen sollte oder nicht. Ein Teil der Trainingsdaten dieses Standardbeispiels ist in Tabelle 2.1 abgebildet.

Outlook	**Temp (° F)**	**Humidity (%)**	**Windy?**	**Class**
sunny	75	70	true	Play
sunny	80	90	true	Dont Play
sunny	85	85	false	Dont Play
sunny	72	95	false	Dont Play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Dont Play
rain	65	70	true	Dont Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Tabelle 2.1: Trainingsdaten T für ein Standardbeispiel
(aus: [13, Seite 18 (2.1.1 An illustration)])

Weil offensichtlich nicht alle Datensätze der Trainingsmenge zu einer Klasse gehören, sondern die beiden Klassen „Play“ und „Dont Play“ aufweisen, wird die Trainingsmenge T in Teilmengen partitioniert, indem der oben beschriebene „divide-and-conquer“-Algorithmus ausgeführt wird. Der Test wird an dieser Stelle auf Basis des Attributs „Outlook“ durchgeführt, weil dieses Kriterium den höchsten Gain-Ratio hat (siehe auch Tabelle 2.2):

$$\begin{aligned} \text{info(T)} &= -9/14 \times \log_2(\underbrace{9/14}_{\text{Play}}) - 5/14 \times \log_2(\underbrace{5/14}_{\text{Dont Play}}) \approx 0,9403 \text{ bits} \\ &= \text{durchschnittlich benötigte Information zur Identifikation} \\ &\quad \text{der Klasse} \end{aligned}$$

$$
\begin{aligned}
\text{info}_{\text{Outlook}}(T) &= +5/14 \times (-2/5 \times \log_2(\overbrace{2/5}^{\text{Play}}) - 3/5 \times \log_2(\overbrace{3/5}^{\text{Dont Play}})) \\
&\quad +4/14 \times (-4/4 \times \log_2(4/4) - 0/4 \times \log_2(0/4)) \\
&\quad +5/14 \times (-3/5 \times \log_2(3/5) - 2/5 \times \log_2(2/5)) \\
&\approx 0,6935 \text{ bits (nach Partitionierung)}
\end{aligned}
$$

$$
\text{gain(Outlook)} = \text{info(T)} - \text{info}_{\text{Outlook}}(T) \approx 0,9403 - 0,6935 = 0,2468 \text{ (bits)}
$$

$$
\begin{aligned}
\text{split\_info(Outlook)} &= -\frac{5}{14} \times \log_2(\underbrace{5/14}_{\frac{|\text{sunny}|}{|T|}}) - \frac{4}{14} \times \log_2(\underbrace{4/14}_{\frac{|\text{overcast}|}{|T|}}) - \frac{5}{14} \times \log_2(\underbrace{5/14}_{\frac{|\text{rain}|}{|T|}}) \\
&\approx 1,5774 \text{ (bits)}
\end{aligned}
$$

$$
\textbf{gain\_ratio(Outlook)} = \frac{\text{gain(Outlook)}}{\text{split\_info(Outlook)}} \approx 0,1564
$$

Position	Gain-Ratio	Attributnummer	Attributbezeichnung
1	0.1564	1	Outlook
2	0.0488	4	Windy?
3	0	3	Humidity (%)
4	0	2	Temp (° F)

Tabelle 2.2: Ranking aller Attribute aus T nach dem Gain-Ratio [ermittelt mit Weka]

Durch die Partitionierung nach „Outlook“ lässt sich die mittlere Gruppierung mit „Outlook = overcast“ und dem Ergebnis „Play“ bereits ausreichend beschreiben. Die erste und dritte Gruppe mit „Outlook = sunny“ bzw. „Outlook = rain“ hingegen nicht. Deswegen wird der Algorithmus rekursiv auf die beiden Teilmengen angewendet. Es findet in der ersten Gruppe eine Fallunterscheidung anhand des Attributs „Humidity“ und den Ergebnissen „Humidity $\leq$ 75“ und „Humidity $>$ 75“ statt. In der zweiten Gruppe erfolgt eine Fallunterscheidung anhand des Attributs „Windy?“ mit den Ergebnissen „windy = true“ und „windy = false“ (siehe dazu das Ranking nach Gain-Ration in Tabellen 2.3 und 2.4). Danach bricht die Rekursion ab, weil je Teilmenge alle Datensätze die gleiche Klasse haben (siehe Fall 1 im Algorithmus).

Position	Gain-Ratio	Attributnummer	Attributbezeichnung
1	1	3	Humidity (%)
2	0.0206	4	Windy?
3	0	2	Temp (° F)

Tabelle 2.3: Ranking der verbleibenden Attribute ohne Outook nach dem Gain-Ratio

Position	Gain-Ratio	Attributnummer	Attributbezeichnung
1	1	4	Windy?
2	0	2	Temp (° F)
3	0	3	Humidity (%)

Tabelle 2.4: Ranking der verbleibenden Attribute ohne Outlook nach dem Gain-Ratio

Der resultierende Entscheidungsbaum ist in der Abbildung 2.3 zu sehen. Es ist zu erkennen, dass auf der ersten Hierarchieebene in der Wurzel die Partitionierung nach „Outlook“ erfolgt und auf der zweiten Hierarchieebene für „Outlook = sunny“ und „Outlook = rain“ eine weitere Unterscheidung nach „Humidity“ bzw. „Windy“ nötig ist. In den Blättern ist entsprechend die Klassifikation und die Anzahl an Fällen, die zu dieser Klasse gehören, enthalten.

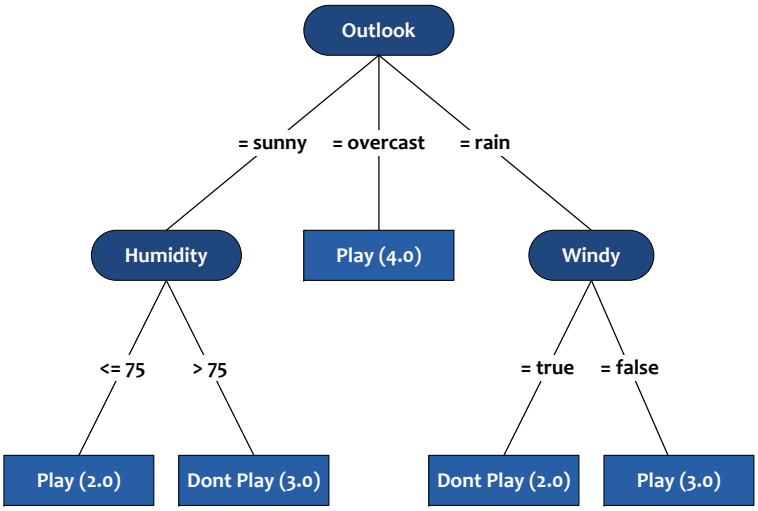


Abbildung 2.3: Entscheidungsbaum für das Standardbeispiel durch den Algorithmus J4.8

3 Analyse

In diesem Kapitel wird das Lokalisierungssystem Nyx analysiert, um potentielle Verbesserungsmöglichkeiten zu entdecken. Dazu werden zunächst die nötigen mathematisch Grundlagen beschrieben.

3.1 Theoretische Grundlagen

Für eine fundierte Analyse von Nyx werden im diesem Abschnitt zunächst die theoretischen Grundlagen beschrieben. Diese stammen vornehmlich aus dem Bereich der Statistik. Grundlegend für das Verständnis ist die Definition der Arbeitshypothese. Eine Arbeitshypothese ist eine widerspruchsfreie, noch zu präzisierende Aussage, die meistens vorläufigen Charakter hat. Die Arbeitshypothese dieser Arbeit quantifiziert die Erkennungsrate des Systems Nyx mit geschätzten 80 bis 85 Prozent. Grundlage dieser Schätzung sind erste einführende Analysen im Vorfeld dieser Arbeit.

Des Weiteren ist die Definition der sogenannten Nullhypothese wichtig. Im Allgemeinen beschreibt diese in der Statistik eine Annahme über die Wahrscheinlichkeitsverteilung einer Zufallsvariablen. Im Rahmen dieser Arbeit entspricht sie der Gegenannahme der Arbeitshypothese, ähnlich einem Beweis durch Widerspruch. Das Pendant zur Nullhypothese stellt die sogenannte Alternativhypothese dar, die in dieser Arbeit die Annahme der Arbeitshypothese beinhaltet. Ziel ist die Bestätigung oder das Verwerfen der Nullhypothese durch Falsifizierung, vorzugsweise durch Ermittlung des (möglichst exakten) Werts der Erkennungsrate.

Ausgehend von Arbeits-, Null- und Alternativhypothese wird also im Rahmen dieser Arbeit zunächst die Fehlerrate bestimmt. Deshalb sei ein Fehler zunächst allgemein definiert als ein von Nyx falsch klassifizierter Port. Diese Definition wird weiter unten noch um einen Aspekt erweitert, der sich aus der Praxis ergibt. Man unterscheidet dabei grundsätzlich die beiden folgenden Fehlerarten:

- **Fehler 1. Art**:
  Der Fehler 1. Art oder auch α-Fehler beschreibt den Fall, dass eine Nullhypothese abgelehnt wird, obwohl sie wahr ist. Er bezeichnet genauer die Wahrscheinlichkeit, dass die Nullhypothese „H_0“ abgelehnt wird, obwohl sie richtig ist. Deshalb heißt der Fehler 1. Art auch „Irrtumswahrscheinlichkeit“. In der Regel akzeptiert man hier einen maximalen Wert von 5% (signifikant) oder 1% (sehr signifikant), der auch als Signifikanzniveau bekannt ist.

- **Fehler 2. Art**:
  Der Fehler 2. Art oder auch β-Fehler beschreibt die Wahrscheinlichkeit, dass eine Nullhypothese nicht abgelehnt wird, obwohl die Alternativhypothese H_1 richtig ist.

	Wahrer Sachverhalt H_0	Wahrer Sachverhalt H_1
Statistischer Test: Entscheidung für die Nullhypothese H_0	$1 - \alpha$	β
Statistischer Test: Entscheidung für die Alternativhypothese H_1	α	$1 - \beta$

Tabelle 3.1: Übersicht über die Klassifikationswahrscheinlichkeiten

Diese Fehlerarten spiegeln sich auch bei der Klassifikation der Ports durch Nyx wider. Wie aus Kapitel 2 bekannt ist, gibt es genau zwei Möglichkeiten zur Klassifizierung eines Ports: Up- bzw. Downlinkport oder Datenport. Damit kann es aufgrund der Existenz der beiden Fehlerarten, deren Wahrscheinlichkeit bei der Klassifikation durch das maschinelle Lernen größer Null ist, zu sogenannten „falsch positiven“ und „falsch negativen“ Klassifikationen kommen:

Falsch positive Klassifikation (engl. „false positives“) Bei einer falsch positiven Klassifikation handelt es sich um den Fehler, den Nyx bzw. der Klassifikationsalgorithmus J4.8 begeht, wenn er Datenports als Up- bzw. Downlinkports klassifiziert.

Falsch negative Klassifikation (engl. „false negatives“) Bei einer falsch negativen Klassifikation hingegen stuft der Algorithmus den Port fälschlicherweise als Datenport ein, obwohl es sich tatsächlich um einen Up- bzw. Downlinkport handelt.

Der Vollständigkeit halber sei an dieser Stelle noch erwähnt, dass korrekt klassifizierte Ports ebenfalls unterschieden werden können. Handelt es sich um einen richtig erkannten Up- bzw. Downlinkport, wird dieser auch als „korrekt positiv“ (engl. „true positive“) bezeichnet. Dem gegenüber wird ein richtig erkannter Datenport als „korrekt negativ“ (engl. „true negative“) gewertet. Dabei bezeichnet die Eigenschaft „positiv“ bzw. „negativ“ – genau wie bei den falsch klassifizierten Fällen – die Perspektive. Bei Nyx ist die Erkennung von Up- und Downlinkports (beispielsweise für die Filterung) entscheidend und deshalb wurde diese Klassifikation als Positiv-Perspektive gewählt. Das Wissen um Datenports ist folglich die Negativ-Perspektive.

Die beschriebenen vier Fälle seien nun noch einmal in Tabelle 3.2 zusammengefasst. In Abhängigkeit vom tatsächlichen Sachverhalt und der Klassifikation, die der Entscheidungsbaum liefert, ergibt sich eine Zuordnung zu genau einem der vier Klassifikationsfälle. Anhand der Tabelle lässt sich offensichtlich auch erkennen, dass die Eindeutigkeit der Zuordnung lediglich von der Unterscheidung (Klassifikation) des betrachteten Ports abhängt, denn der wahre Sachverhalt ist stets gegeben und fest.

	Wahrer Sachverhalt: **Up- bzw. Downlinkport** (positiv)	Wahrer Sachverhalt: **Datenport** (negativ)
Entscheidungsbaum liefert positive Klassifikation **Up- bzw. Downlinkport**	(Anzahl der) **Korrekt-Positiven** = (Number of) **True Positives** [kurz: (N)TP]	(Anzahl der) **Falsch-Positiven** = (Number of) **False Positives** [kurz: (N)FP]
Entscheidungsbaum liefert negative Klassifikation **Datenport**	(Anzahl der) **Falsch-Negativen** = (Number of) **False Negatives** [kurz: (N)FN]	(Anzahl der) **Korrekt-Negativen** = (Number of) **True Negatives** [kurz: (N)TN]

Tabelle 3.2: Übersicht über die Klassifikationsfälle bei Nyx

Aus den bisherigen Erkenntnissen und der Gruppeneinteilung aus Abbildung 3.2 lassen sich wiederum weitere sehr wichtige Verhältnisse ableiten. Dafür bezeichne „NTP“ (für engl. „number of true positives“) die Anzahl der Korrekt-Positiven, „NFN“ (für engl. „number of false negatives“) die Anzahl der Falsch-Negativen usw. Damit erhält man leicht die Definitionen für die relevanten Verhältnisse:

$$FP = \frac{NFP}{NFP + NTN} = \beta \tag{3.1}$$

$$FN = \frac{NFN}{NTP + NFN} = \alpha \tag{3.2}$$

$$TP = \frac{NTP}{NTP + NFN} = 1 - \alpha \tag{3.3}$$

$$TN = \frac{NTN}{NTN + NFP} = 1 - \beta \tag{3.4}$$

Für die Korrektheit (engl. „accuracy“, AC), die die Anzahl der korrekt erkannten Fälle im Verhältnis zur Zahl aller aufgetretenen Fälle beschreibt, ergibt sich dann:

$$AC = \frac{NTP + NTN}{NTP + NFP + NFN + NTN} \tag{3.5}$$

$$\text{bzw. } AC = TP = \frac{NTP}{NTP + NFN} = 1 - \alpha, \tag{3.6}$$

wenn Datenports nicht berücksichtigt werden sollen.

Diese Definitionen und Verhältnisse gelten unter der Voraussetzung, dass man alle Werte direkt überprüfen und die Zugehörigkeit eines bestimmten Falls zu einer der vier Klassifikationsgruppen festgestellt werden kann. Im Fall der Datenports ist das jedoch nicht gegeben. Deshalb wird bereits an dieser Stelle ein Korrektiv (Ausgleichsfehler, engl. „adjustment error“, AE) eingeführt. Die Anzahl der richtig erkannten Datenports wird dazu prozentual durch die

Gewichtung *dl* (für engl. „dataport loading“) in zwei Teile zerlegt, wobei aV für „adjustment Value“ und aNTN für „adjusted number of true negatives“ steht:

$$aV = NTN * dl \tag{3.7}$$
$$aNTN = NTN - (NTN * dl) \tag{3.8}$$
$$\text{sodass: } NTN = aNTN + aV \tag{3.9}$$

Damit ändert sich die Definition der Korrekt-Negativ-Rate, es entsteht ein Ausgleichsfehler und die Korrektheit muss entsprechend angepasst werden.

$$TN = \frac{aNTN}{NTN + NFP} = 1 - \beta \tag{3.10}$$
$$AE = \frac{aV}{NFP + NTN} \tag{3.11}$$
$$AC = \frac{NTP + aNTN}{NTP + NFP + NFN + aNTN + AE} \tag{3.12}$$

Mit diesen Werten lässt sich im Weiteren das System Nyx analysieren. Wie die Analyse methodisch aufgebaut ist, wird im nächsten Abschnitt genauer beschrieben.

3.2 Möglichkeiten zur Analyse

Es gibt prinzipiell verschiedene potenzielle Ansätze zur Überprüfung der Klassifikation eines Ports. So kann man beispielsweise manuelle und automatisierte Verfahren unterscheiden. Gleich ist dabei allen Methoden, dass sie versuchen, die Klassifikation mit der „Realität“ zu vergleichen. Inwieweit dies jedoch überhaupt möglich ist, zeigen die folgenden Methoden, die im Rahmen dieser Arbeit untersucht wurden.

3.2.1 Manuelle Überprüfung via Command Line Interface

Die erste Möglichkeit zur Überprüfung der Klassifikation eines Ports stellt die sogenannte Kommandozeilenschnittstelle (engl. „command line interface“, CLI) eines Switches zur Verfügung. Das CLI ist eine Schnittstelle mit allen Funktionen zur Ermittlung von Statusinformationen und zur Konfiguration der Switches. Diese sind über telnet (kurz für „telecommunication network“), einem weit verbreiteten Netzwerkprotokoll, erreichbar. Zum Schutz der Informationen und Funktionen ist der eigentlichen Kommandozeilenumgebung eine passwortbasierte Authentifikation vorgelagert. Dadurch können auch bestimmte Teile der CLI-Funktionalität benutzerspezifisch freigegeben bzw. gesperrt werden, sodass lediglich vordefinierte Funktionen genutzt werden können.

Zur Anzeige des Switch-Menüs wird der Befehl „`menu`“ verwendet. Im Hauptmenu selbst hat man meist die Möglichkeit, sich die diversen Statusanzeigen und Zähler (Counter) zum Datenverkehr etc. anzeigen zu lassen. Beispiel 3.1 zeigt exemplarisch die Schritte, die nötig sind, um sich sämtliche MAC-Adressen am Port 1 eines Switches anzeigen zu lassen. Zunächst

muss im Hauptmenü des Switches das Menü „Status und Counters...“ aufgerufen werden, anschließend die Option „Port Address Table“ und als Parameter die Portbezeichnung, hier „A1“ für Port 1. Als Ergebnis erhält man eine sortierte Liste von MAC-Adressen, die an diesem Port des Switches sichtbar sind.

Über die Kommandozeilenschnittstelle ist es offensichtlich möglich, viele verschiedene Informationen zu erhalten. Allerdings ermöglichen sie für sich gesehen keine Überprüfung der Klassifikation. Eine Unterscheidung von Inter-Switch- und Endgeräte-Verbindungen allein über das CLI ist nicht möglich, weil in den Switch-Statuslisten und Counter nicht festgehalten wird, ob es sich bei einem bestimmten Port um einen Downlinkport oder einen Datenport handelt. Das wäre nötig, weil die MAC-Adresse an beiden Portarten auf dem gesamten Kommunikationspfad erscheint. Auch wenn die Liste der MAC-Adressen (Port Adress Table) beispielsweise nur einen Eintrag beinhaltet, kann trotzdem über CLI nicht eindeutig bestimmt werden, ob es sich um einen Downlinkport zu einer weiteren Netzkomponente oder um den Datenport vom Edge-Switch zum Endgerät handelt. Dazu müsste man anhand der MAC-Adresse entscheiden können, um welche Art von Gerät es sich handelt. Dies ist nicht eindeutig möglich, weil MAC-Adressen zwar eine Herstelleridentifikation enthalten können, aber nicht zwingend müssen und zudem beliebig verändert werden können.

Zusätzliche Informationen aus dem Rapid Spanning Tree Protokoll (RSTP), das von einigen Switches unterstützt wird, können ebenfalls nicht ausschließlich verwendet werden. Dieses Protokoll ist nämlich nicht auf allen Switches aktiviert. Damit bleiben die Daten, die auch Nyx über SNMP bezieht. Die Überprüfung kann also nicht auf Basis des CLI erfolgen.

3.2.2 Manuelle Überprüfung via Nyx-Datenbank und SQL-Query

Der im vorherigen Abschnitt beschriebene Ansatz, eine manuelle Überprüfung der Klassifikation durchzuführen, indem die Informationen zur Beurteilung direkt von Switch über die Kommandozeilenschnittstelle bezogen werden, ist also nicht geeignet. Die relevanten Daten, wie zum Beispiel die sichtbaren MAC-Adressen an einem bestimmten Port, sind zudem ohnehin in den Datenbank-Tabellen von Nyx vorhanden, denn diese werden über SNMP regelmäßig in Echtzeit abgefragt.

In diesem Abschnitt wird deshalb untersucht, ob eine Unterscheidung von Inter-Switch- und Endgeräte-Verbindung auf logischer Basis erfolgen kann. Dazu wird prinzipiell versucht, die Inter-Switch-Verbindungen zu erkennen. Eine Verbindung ist genau dann eine Inter-Switch-Verbindung, wenn beide Ports bzw. deren korrelierende MAC-Adresse zu bekannten, managebaren Netzkomponenten gehören.

Graphentheoretisch kann man sich eine Inter-Switch-Verbindung als innere Kante vorstellen, d.h. es handelt sich um eine Kante zwischen zwei inneren Knoten (siehe Abbildung 3.2). Im Gegensatz dazu sind Endgeräte-Verbindungen – also vom Edge-Switch zum Endgerät – graphentheoretisch immer Kanten zwischen einem inneren Knoten und einem Blatt-Knoten. Netzkomponenten (im Wesentlichen Switches) sind also in diesem Bild immer innere Knoten und gehören zum verwaltbaren (managebaren) Teil des Netzes. An diese innere Struktur sind (am Netzrand) Endgeräte angeschlossen, die damit folgerichtig Blätter des Baums darstellen. Der Edge-Switch selbst ist verwaltbar und gehört damit noch zur inneren Struktur, denn würde man ihn als Blatt werten, wäre in diesem Bild keine Verwaltbarkeit möglich, da im Bild lediglich der innere Teil managebar sein soll.

Beispiel 3.1 (Beispiel für die Ermittlung der MAC-Adressen für Port A1)

```
telnet swg1-0w5.net.lrz-muenchen.de
```

```
SWG1-0W5>menu
```

```
SWG1-0W5                                                [date] [time]
====================- TELNET - OPERATOR MODE -====================

                                Main Menu

     1. Status and Counters...
     2. Event Log
     3. Command Line (CLI)
     0. Logout

Provides the menu to display configuration, status, and counters.
To select menu item, press item number, or highlight item and press
<Enter>.
```

```
SWG1-0W5                                                [date] [time]
====================- TELNET - OPERATOR MODE -====================

           Status and Counters - Port Address Table - Port A1

     MAC Address
     ---------
     00022d-a53ecc
     000352-0373f3
     000352-03d5b8
     000352-03ea04
     000352-0a0a27
     000352-0a53cf
     000352-0a936b
     000352-0a9528
     000352-0a9529
     000423-6fdcff
     000476-cd3fc6
     000476-e8b7bd

   Actions-> Back Search Next page Prev page Help

Return to previous screen.
Use up/down arrow keys to scroll to other entries, left/right arrow keys
to change action selection, and <Enter> to execute action.
```

Abbildung 3.1: Port-Address-Table eines Switches

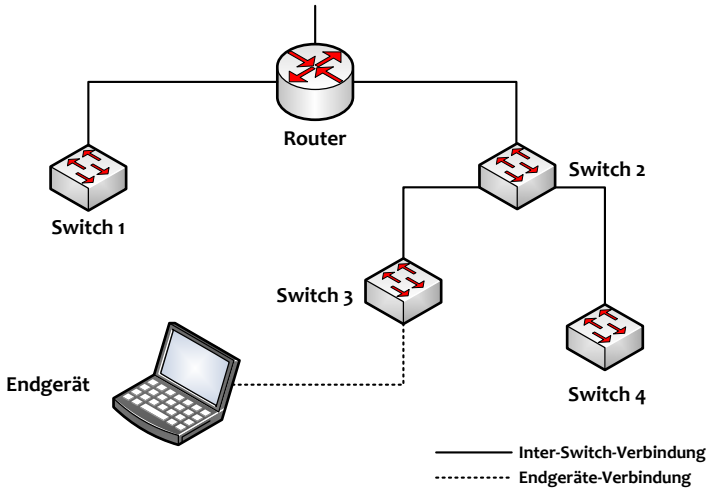


Abbildung 3.2: Veranschaulichung von Inter-Switch- und Endgeräte-Verbindungen

Ein solcher Ansatz ließe sich theoretisch mit SQL (Structured Query Language) realisieren. Praktisch ist er aber wegen der großen Zahl an Ausnahmen und der steigenden Komplexität unbrauchbar. Das haben Versuche im Rahmen dieser Arbeit (und zuvor) gezeigt. Das im Szenario beschriebene Münchner Wissenschaftsnetz ist hierfür schlicht zu komplex. Deshalb sei an dieser Stelle lediglich das prinzipielle Vorgehen anhand des Switches aus Beispiel 3.1 skizziert:

Listing 3.1: SQL-Statement zur Ermittlung der verbundenen Netzkomponenten für den Switch aus Beispiel 3.1 mit der ID „HPJ4865ASG31361051“ an Port „1“

```
SELECT DISTINCT
  ifmac. * ,
  device_physicalmac.device AS connectedDevice

FROM
  ( SELECT DISTINCT device, interfaceid, ifindex, mac
    FROM mac_physical
    WHERE interfaceid LIKE "HPJ4865ASG313610511"
  ) AS ifmac,

  device_physicalmac

WHERE device_physicalmac.physicalmac = ifmac.mac
```

Zuerst werden alle MAC-Adressen zu bekannten, verwaltbaren Netzkomponenten am betrachteten Port bestimmt. Dazu verwendet man die Tabelle „mac_physical“, in der diese MAC-Adressen für alle Geräte (engl. „devices“) bzw. genauer für alle Ports („interfaces“) gespeichert sind. In Listing 3.1 werden exemplarisch die MAC-Adressen am Port „1“ bzw. „A1“ am Gerät „HPJ4865ASG31361051“ bestimmt. Diese Anfrage erfolgt als inneres SQL-

Statement und dient anschließend als erste Datenbasis für die äußere Anfrage. Das Resultat für den Beispiel-Switch ist die MAC-Adresse „00:D0:03:2E:B8:00“.

Als zweite Datenbasis wird die Tabelle „device_physicalmac“ verwendet, in der alle MAC-Adressen einer bekannten, verwaltbaren Netzkomponente gespeichert sind. Die Kombination der beiden Tabellen mit der WHERE-Bedingung ergibt dann diejenigen Netzkomponenten, die direkt oder mittelbar mit diesem Port verbunden sind. Wenn die Liste nicht leer ist, handelt es sich um einen Uplink- oder Downlinkport, ansonsten um einen Datenport. Die Tabellen 3.4 bis 3.7 zeigen die Ergebnisse für den Switch aus Beispiel 3.1 an verschiedenen Ports sowie für den Switch „HPJ4865ASG31361051“, dessen Uplinkport „A1“ betrachtet wurde.

Die Tabelle 3.3 zeigt den Switch aus Beispiel 3.1 und alle verbundenen Netzkomponenten. Diese Tabelle stammt aus der LRZ Netzdokumentation, die zwar nicht in jedem Fall korrekt sein muss, aber hier als sicherer Vergleichswert für die Ergebnisse der SQL-Anfrage dienen soll. Außerdem zeigt die Tabelle die Daten für einen weiteren Switch, der seinerseits über Port 1 an den ersten Switch angeschlossen ist.

In den folgenden drei Tabellen wird dann das Ergebnis des SQL-Statement für den ersten Switch und den jeweiligen Port (1, 2, und 120) angezeigt. Die ersten beiden Ports sind offensichtlich Up- bzw. Downlinkports, denn die Liste ist nicht leer, sondern enthält eine verbundene Netzkomponente. Der letzte Port ist hingegen ein Datenport, da die Liste leer ist. Die letzte Tabelle 3.7 zeigt zusätzlich das Ergebnis aus Sicht des an Port 2 angeschlossenen zweiten Switches. Die Liste der Einträge enthält den Switch aus Beispiel 3.1, den Router (beginnend mit „SCA“) und weitere Netzkomponenten. Die vorherigen Ergebnisse werden dadurch bestätigt.

Die Idee dieses Ansatzes hat gewisse Ähnlichkeiten zum bereits verwendeten Kriterium „component_mac_discovered“, das in Abschnitt 2.2.3 beschrieben wurde, erlaubt aber wesentlich präzisere Aussagen als dieses Kriterium. Das SQL-Statement zeigt nämlich zusätzlich das verbundene Gerät, sodass theoretisch eine weitergehende Erkennung der Topologie erfolgen kann als dies bei dem von Nyx verwendeten Kriterium der Fall ist, das lediglich die erfolgreiche Entdeckung einer Netzkomponente auf Basis der MAC-Adresse festhält.

3.2.3 Automatisierter Abgleich

Die beiden bisherigen Ansätze haben gezeigt, dass eine manuelle Überprüfung der Klassifikation von Ports nur sehr beschränkt möglich ist, weil zum einen die nötigen Netzstrukturen nicht abgebildet und dadurch nicht abfragbar sind und zum anderen die Komplexität des Netzes im Szenario sehr hoch ist. Aus diesem Grund wird nun versucht, die Überprüfung der Port-Klassifikation zu automatisieren. Im Wesentlichen werden dazu automatisiert die Eigenschaften des Geräts und des Ports analysiert, um auf Basis dessen eine Klassifikation durchzuführen. Diese wird anschließend mit der Klassifikation, die durch Nyx bzw. durch den maschinellen Lernalgorithmus erfolgt ist, abgeglichen und das Ergebnis des Vergleichs zur Bestimmung der Fehler- bzw. Erkennungsrate genutzt. Zur Durchführung dieses automatischen Abgleichs wurde das Perl-Skript „Eos“[1] entwickelt, das im folgenden Abschnitt genauer beschrieben wird.

[1] Eos ist in der griechischen Mythologie die Göttin der Morgenröte, die das erste Licht am neuen Tag bringt. Die Wahl dieses Namens erfolgte in Anlehnung an Nyx, die personifizierte Nacht, und soll das Ziel dieser Arbeit, durch Analyse der Leistung von Nyx „Licht ins Dunkel“ zu bringen, verdeutlichen.

Switch:	HPJ4865ASG31361051	**verbunden mit**	
Portbezeichnung	**Portnummer**	**interfaceid**	**sysname**
A1	1	SCA044403RU	csr1-kw5 <7/3>
A2	2	HPJ4865ASG31361050	swg1-kw5 <A1>
E24	120	(keine)	ap01-0w5 <E1>

Switch:	HPJ4865ASG31361050	**verbunden mit**	
Portbezeichnung	**Portnummer**	**interfaceid**	**sysname**
A1	1	HPJ4865ASG31361051	swg1-0w5 <A2>
E24	120	(keine)	smw40002 <8>

Tabelle 3.3: Ausschnitt des Netzbereichs beim Switch aus Beispiel 3.1

device	**interfaceid**	**ifindex**	**mac**	**connectedDevice**
HPJ4865ASG31361051	HPJ4865ASG313610511	1	00:D0:03:2E:B8:00	SCA044403RU

Tabelle 3.4: Ergebnis für den Switch aus Beispiel 3.1 an Port „1“

device	**interfaceid**	**ifindex**	**mac**	**connectedDevice**
HPJ4865ASG31361051	HPJ4865ASG313610512	2	00:0A:57:3F:A2:08	HPJ4865ASG31361050

Tabelle 3.5: Ergebnis für den Switch aus Beispiel 3.1 an Port „2“

device	**interfaceid**	**ifindex**	**mac**	**connectedDevice**

Tabelle 3.6: Ergebnis für den Switch aus Beispiel 3.1 an Port „120“ (Datenport)

device	**interfaceid**	**ifindex**	**mac**	**connectedDevice**
HPJ4865ASG31361050	HPJ4865ASG313610501	1	00:04:EA:7E:B9:04	HPJ4865ASG22760801
HPJ4865ASG31361050	HPJ4865ASG313610501	1	00:0A:57:3F:98:07	**HPJ4865ASG31361051**
HPJ4865ASG31361050	HPJ4865ASG313610501	1	00:0A:57:6D:95:05	HPJ4865ASG315MF0FL
HPJ4865ASG31361050	HPJ4865ASG313610501	1	00:15:60:81:25:01	HPJ4865ASG613MF009
HPJ4865ASG31361050	HPJ4865ASG313610501	1	00:04:EA:3B:6E:04	HPJ4865ASG22360805
HPJ4865ASG31361050	HPJ4865ASG313610501	1	00:D0:03:2E:B8:00	**SCA044403RU**

Tabelle 3.7: Ergebnis für den Switch „HPJ4865ASG31361051“ an Port „1“

3.3 Eos: Ein Ansatz zur automatisierten Analyse von Nyx

3.3.1 Einführung

Ziel dieses Ansatzes ist die automatisierte Überprüfung der Klassifikationen von Ports. Die Ausgangssituation für das Vorhaben, eine automatisierte Analyse durchzuführen, ist das Wissen um die Notwendigkeit einer automatisierten Lösung aufgrund des Scheiterns der manuellen Ansätze.

Die für diesen Ansatz zur Verfügung stehenden Mittel beschränken sich vor allem auf die Datenbanken von Nyx sowie die LRZ Netzdokumentation. Aufgabe der Netzdokumentation ist die Erfassung, Speicherung und Darstellung eines möglichst aktuellen Zustands des durch das LRZ verwalteten Netzes. Da dieses System allerdings manuell gepflegt wird, sind Fehler nicht ganz auszuschließen. Außerdem ist die Aktualität der Daten nicht unmittelbar ersichtlich, sodass Abweichungen von der Realität möglich sind. In diesem Zusammenhang sei ein grundsätzliches Problem beschrieben, das im Folgenden als „Unentscheidbarkeitsproblem" bezeichnet wird. Dieses beschreibt die Problematik, einen Sachverhalt sicher beurteilen zu müssen und dafür aber lediglich unsichere Daten zur Verfügung zu haben. Das Problem lässt sich anhand eines Beispiels nachvollziehen.

Beispiel 3.2 (Beispiel für das Unentscheidbarkeitsproblem) *Betrachtet wird die Netzkomponente* `SWZ3-2WR`*, ein Zentralswitch für das Linux-Cluster im LRZ Rechnerwürfel. Der Port (A)4 an diesem Switch (interfaceid: „*`HPJ8698ASG640SV02S4`*") ist laut maschinellem Lernalgorithmus ein* ***Up- bzw. Downlinkport****. Die Netzdokumentation enthält allerdings keinen Eintrag, was auf einen* ***Datenport*** *hinweist. Schaut man sich die durch SNMP-Abfrage erhaltenen Daten an, so weist der Port (A)4 durchschnittlich 34 registrierte MAC-Adressen auf. Diese Zahl erklärt die Klassifikation des maschinellen Lernalgorithmus, der in jedem Fall einen Port mit durchschnittlich mehr als zehn MAC-Adressen als* ***Up- bzw. Downlink*** *klassifiziert. Es gibt also jeweils einen sehr guten Grund für beide Klassifikationen: Die SNMP-Daten, die in Echtzeit abgefragt werden, und der fehlende Eintrag in der sorgfältig gepflegten Netzdokumentation.*

Es stellt sich also im Rahmen des Unentscheidbarkeitsproblems die Frage, wie sicher die Daten sind, d.h. inwieweit sie der Realität entsprechen. Die Quellen müssen also bewertet bzw. gewichtet werden, um eine Rangfolge angeben zu können, auf Basis derer eine Gewichtung erfolgen kann.

Im Beispiel 3.2 hat die Netzdokumentation die Realität korrekt widergespiegelt. Im Allgemeinen ist dies allerdings nicht zwingend der Fall. Trotzdem sei an dieser Stelle die Korrektheit der Netzdokumentation postuliert, die im Folgenden vorausgesetzt wird, weil nur so ein automatisierter Abgleich Sinn hat. Dieses Postulat bedeutet, dass die Informationen der Netzdokumentationen stets höheres Gewicht haben als die des maschinellen Lernalgorithmus. Nur so sind Aussagen für Fälle möglich, die nicht eindeutig sind, sondern widersprüchliche Klassifikationen aufweisen. Es ist nämlich nicht möglich, für jede zweifelhafte Klassifikation eine explizite oder implizite Sichtung (durch Dritte) durchzuführen. Der Grund dafür ist die große Zahl an Ports (> 65.000) und Netzkomponenten (> 1000) im Münchner Wissenschaftsnetz (vgl. Tabelle 1.1 aus dem Szenario).

Ausgehend von diesen zur Verfügung stehenden Mitteln und den technischen Gegebenheiten soll nun das Skript „Eos“ im ersten Schritt anhand der Informationen aus der Netzdokumentation die Eigenschaften eines Ports ermitteln und diesen anhand derer einer der beiden Klassen, Up- bzw. Downlinkport oder Datenport, zuordnen. Im zweiten Schritt soll diese Klassifikation dann mit derjenigen, die durch den maschinellen Lernalgorithmus ermittelt worden ist, verglichen werden. Bei den Vergleichen können folgende **Ergebnisklassen** auftreten:

RECOGNIZED: Der Switch sowie der Port wurden in der Netzdokumentation gefunden und sowohl Nyx als auch die Netzdokumentation zeichnen den Port als Up- bzw. Downlink aus.

FALSE NEGATIVE: Der Switch sowie der Port wurden in der Netzdokumentation gefunden. Nyx klassifiziert den Port als Datenport, obwohl er laut Netzdokumentation ein Up- bzw. Downlinkport ist.

HEURISTICALLY RECOGNIZED: Der Switch wurde zwar in der Netzdokumentation gefunden, der Port, der von Nyx als Up- bzw. Downlink klassifiziert wurde, jedoch nicht. Außerdem hat die Heuristik, die im Nyx eingebaut ist, um beispielsweise sogenannte Bridging-Firewalls zu erkennen, den Port als Up- bzw. Downlinkport erkannt. Es handelt sich also nicht um einen Klassifikationsfehler, weil in der Netzdokumentation solche Informationen nicht gespeichert werden, da sie zumeist nicht bekannt sind.

FALSE POSITIVE: Der Switch sowie der Port wurden in der Netzdokumentation gefunden. Nyx klassifiziert den Port als Up- bzw. Downlinkport, obwohl er laut Netzdokumentation ein Datenport ist.

DATAPORT: Der Switch wurde in der Netzdokumentation gefunden und der Port wird gar nicht oder als Datenport zu einem Endgerät deklariert. Zusätzlich klassifiziert der maschinelle Lernalgorithmus den Port korrekt als Datenport.

NO SWITCH: In diesem Fall wurde der Switch nicht in der Netzdokumentation gefunden. Ein automatisierter Abgleich kann hier nicht erfolgen.

3.3.2 Implementierung

Eos ist ein Analyse-Skript zur automatisierten Überprüfung der Portklassifikation, die durch den maschinellen Lernalgorithmus erfolgt. Es ist vollständig in Perl geschrieben, da es wichtige Datenstrukturen sehr schnell und unkompliziert zur Verfügung stellt. Dazu gehören unter anderem Listen und vor allem assoziative Arrays, d.h. Arrays mit nicht-numerischen Schlüsseln, die bei der Implementierung von Eos eine entscheidende Rolle spielen.

Das Skript Eos besteht aus folgenden Teilabschnitten:

1. Konfiguration
2. Prüfung und Auswertung der Parameter
3. Verbindungsaufbau zu den Datenbanken

4. Auslesen der relevanten Daten aus Nyx
5. Auslesen der relevanten Daten aus der Netzdokumentation
6. Überprüfung der Klassifikation
   a) Erstellung von Ausgabedateien
   b) Überprüfung
7. Statistische Berechnungen (siehe auch Abschnitt 3.3.3)
   a) False Positive und False Negative
   b) True Positive und True Negative
   c) Anpassungsfehler („adjustment error“)
   d) Korrektheit („accuracy“), Konfidenzintervall und Sicherheit
8. Ausgabe

Die Konfiguration von Eos beinhaltet die Parameter für den Zugriff auf die beiden Datenbanken von Nyx und der Netzdokumentation. Im Wesentlichen sind dies die IP-Adresse, der Port, die Zugangsdaten sowie der Datenbankname. Diese können lediglich im Quellcode verändert werden, weil angenommen wird, dass diese Daten relativ stabil sind, d.h. selten verändert werden müssen. Der Einsatz einer Konfigurationsdatei bringt an dieser Stelle offensichtlich keine Vorteile, da der Quellcode von Eos direkt bearbeitet werden kann.

Die Parameterprüfung stellt die zweite Komponente von Eos dar. Über die Parameter ist es dem Benutzer möglich, die Analyse zu beeinflussen. Eos bietet dafür die folgenden **Parameter**:

-samples=<integer> bzw. -s=<integer> Über diesen Parameter kann die maximale Anzahl der zu testenden Ports angegeben werden. Diese Zahl gibt also die maximale Größe der Stichprobe (siehe Abschnitt 3.3.3) an und stellt damit eine obere Grenze dar. Der Grund dafür ist, dass man nicht bei allen Elementen dieser Stichprobe von deren Eignung bzw. Gültigkeit ausgehen kann. Ein Stichprobenelement (aus Nyx) ist genau dann ungültig, wenn man es nicht mit Elementen aus der Netzdokumentation vergleichen kann.

-dataport_loading=<float> bzw. -dl=<float> Der Parameter „dataport_loading“ gibt eine Gewichtung für die residual erkannten Datenports an. Ohne diese Gewichtung würde man implizit davon ausgehen, dass die Datenports nicht residual, sondern tatsächlich erkannt werden können. Dies ist jedoch nicht der Fall, da die Netzdokumentation nur Informationen zu wenigen bekannten Datenports beinhaltet. Der ermittelte residuale Wert der Dataports beinhaltet also immer einen gewissen inherenten Fehler, der im Skript durch einen „Anpassungsfehler“ beschrieben wird.

-heuristics bzw -h Dieser Parameter kontrolliert, ob die Klassifikation eines Interfaces als Datenport beibehalten wird, obwohl durch einen unabhängigen heuristischen Filter das Verhalten eines Up- bzw. Downlinkports festgestellt wurde. Wird diese Option ausgewählt, werden alle Datenports, bei denen die Heuristik anschlägt, zu Up- bzw. Downlinkports umklassifiziert. Dieser Parameter hat nur indirekt mit der Einstufung als „Heuristically Recognized“ zu tun, für die noch weitere Kriterien erfüllt sein müssen als nur das Ergebnis der Heuristik.

-accesspoints bzw. -ap Über diesen Parameter kann beeinflusst werden, wie Access Points bewertet werden. Access Point sind zwar eigentlich Netzkomponenten, werden aber aktuell aus technischen Gründen von Nyx als Endgeräte behandelt. Mit diesem Parameter kann man deshalb kontrollieren, wie Eos Access Points in seiner Auswertung erfasst. Ist er vorhanden, werden Ports von Access Points als Up- bzw. Downlinkports gewertet.

-key Wenn angegeben, wird zusätzlich zum Ergebnis eine Legende ausgegeben.

-keyonly Es wird nur die Legende ausgegeben ohne eine Analyse durchzuführen.

-help Zeigt Hilfestellungen zur Nutzung von Eos.

Anschließend an die Überprüfung und Auswertung der Parameter erfolgt der Verbindungsaufbau zu den Datenbanken. Ist diese erfolgreich hergestellt, werden die Daten von Nyx und der Netzdokumentation ausgelesen. Dazu werden die folgenden Datenstrukturen erzeugt:

Listing 3.2: Datenstrukturen für Nyx

```
my @interfaceids;       # list of ids (array)
my %ifnames;            # interfaceid -> ifdescr (hash)
my %devices;            # interfaceid -> sysname (hash)
my %reversed_devices;   # sysname -> interfaceid (hash)
my %classify;           # interfaceid -> updownlink (hash)
```

Listing 3.3: Datenstrukturen für die Netzdokumentation

```
# Komponente: id->alias (hash)
my %nd_devices;

# PhyPort: (if)id->Bezeichnung (hash)
my %nd_bezeichnungen;

# phyport.id->phyport.komponente_id (hash)
my %nd_interfaces;

# phyport.id->phyport.leitung_id (hash)
my %nd_connected_via_line;

# phyport.id->connected_phyport_id (hash)
my %nd_connected_to_if;

# phyport.id->Komponente.alias
my %nd_device_alias_from_portid;

# list of netzdoku ids (array)
my @nd_ids;
```

Die Erzeugung dieser Datenstrukturen erfolgt im Wesentlichen kanonisch: Es werden zunächst entsprechende SQL-Anfrage an die Datenbanken gestellt und die Rückgabe (iterativ über alle Elemente) in den Strukturen gespeichert. Dabei erfolgt in Abhängigkeit der Programmparameter eine gewisse Selektion bzw. Filterung der Rückgaben. Bei Aktivierung des

Parameters „heuristical“ wird beispielsweise die heuristische der maschinellen Klassifikation durch den Lernalgorithmus vorgezogen.

Als Besonderheit sei an dieser Stelle die Konstruktion der beiden Hashes „devices“ und „reversed_devices“ erwähnt, für die gilt:

$$\text{interfaceid} = \text{devices (reversed\_devices (interfaceid))} \tag{3.13}$$

$$\text{bzw. sysname} = \text{reversed\_devices (devices (sysname))} \tag{3.14}$$

In Schritt 6 findet schließlich die Überprüfung der Klassifikation statt. Dazu wird zunächst ein Ausgabeverzeichnis (mit Datums- und Uhrzeitangabe im Namen) für die Textdateien erstellt, die die Ergebnisse je nach Vergleichsresultat (siehe Abschnitt 3.3.1, Ergebnisklassen) beinhalten. Danach wird die Überprüfung iterativ über alle ausgelesenen Interfaces durchgeführt.

Der erste Schritt dabei ist eine Basis-Überprüfung, die sicherstellt, dass alle notwendigen Informationen für eine Überprüfung vorhanden sind und es sich nicht um einen Router handelt. Diese bleiben nämlich sowohl bei Nyx als auch bei Eos unberücksichtigt, weil es sich bei Router-Ports per Definition nie um Datenports handelt. Außerdem werden Ports, bei deren Gerät der Name (sysname) leer oder deren Portbeschreibung (engl. „interface description“, kurz: „ifdescr“) länger als drei Buchstaben ist, als ungültig gewertet, da sie nicht unter allen Umständen für einen Abgleich geeignet sind. Die zweite Bedingung ist nötig, weil teilweise unbrauchbare Daten (Kommentare, etc.) im Beschreibungsfeld stehen, sodass ebenfalls ein Vergleich mit der Netzdokumentation nicht durchgeführt werden kann. Wenn die Portbeschreibung hingegen maximal drei Buchstaben bzw. Zahlen enthält, wird davon ausgegangen, dass sie gültig und für eine Überprüfung geeignet ist. Die eigentliche Klassifikationsüberprüfung erfolgt anschließend nach dem Schaubild 3.3. Um dabei unnötige Schleifendurchläufe zu verhindern, wurde eine spezielle Variable eingeführt, sodass die Schleifendurchläufe abgebrochen werden, sobald ein Port durch Eos in eine Ergebnisklasse eingeordnet worden ist.

Sind alle gültigen Ports überprüft worden, werden alle Ausgabedateien (bis auf die Datei zur Speicherung des Ergebnisses) geschlossen und die statistischen Berechnungen durchgeführt (siehe Eos-Teilkomponenten, Punkt 7, Seite 32). Dazu wird zunächst der Anpassungswert („adjustment Value“) mittels Gewichtungsfaktor für die Datenporterkennung ermittelt und von der Anzahl der erkannten Datenports subtrahiert, um die angepasste Zahl der Datenports zu erhalten. Ist die Gewichtung an dieser Stelle 1, d.h. man geht davon aus, dass die Datenports von Nyx alle korrekt erkannt werden, dann ist der Anpassungsfehler offensichtlich gleich Null und die Zahl der Datenports (NTN) ist gleich der Zahl der angepassten Datenports (aNTN). Diese Berechnungen entsprechen den Definitionen 3.7 bis 3.9 (Seite 24) und finden sich in Listing 3.4 in den ersten beiden Anweisungen wieder.

Anschließend wird die Zahl der berücksichtigten Fälle als Summe der Anzahl an Korrekt-Positiven (`$nyx_updownlinkcount`), Falsch-Negativen (`$nyx_false_negative`) sowie Falsch-Positiven (`$nyx_false_positives`), und Korrekt-Negativen (`$nyx_dataport` zusammen mit `$adjustmentValue`) ermittelt. Die Anzahl der gültigen Fälle ist weiter gleich der Anzahl der Fälle, die den Basis-Test (siehe oben) erfolgreich durchlaufen haben und demnach die ungültigen Fälle der residuale Anteil, wenn man die Zahl der gültigen Fälle von der Zahl aller Fälle (= Parameter „`-samples=<integer>`“) subtrahiert. Diese Berechnungen spiegeln sich in Listing 3.4 in den Anweisungen wider, die mit „`,,$samples`“ beginnen.

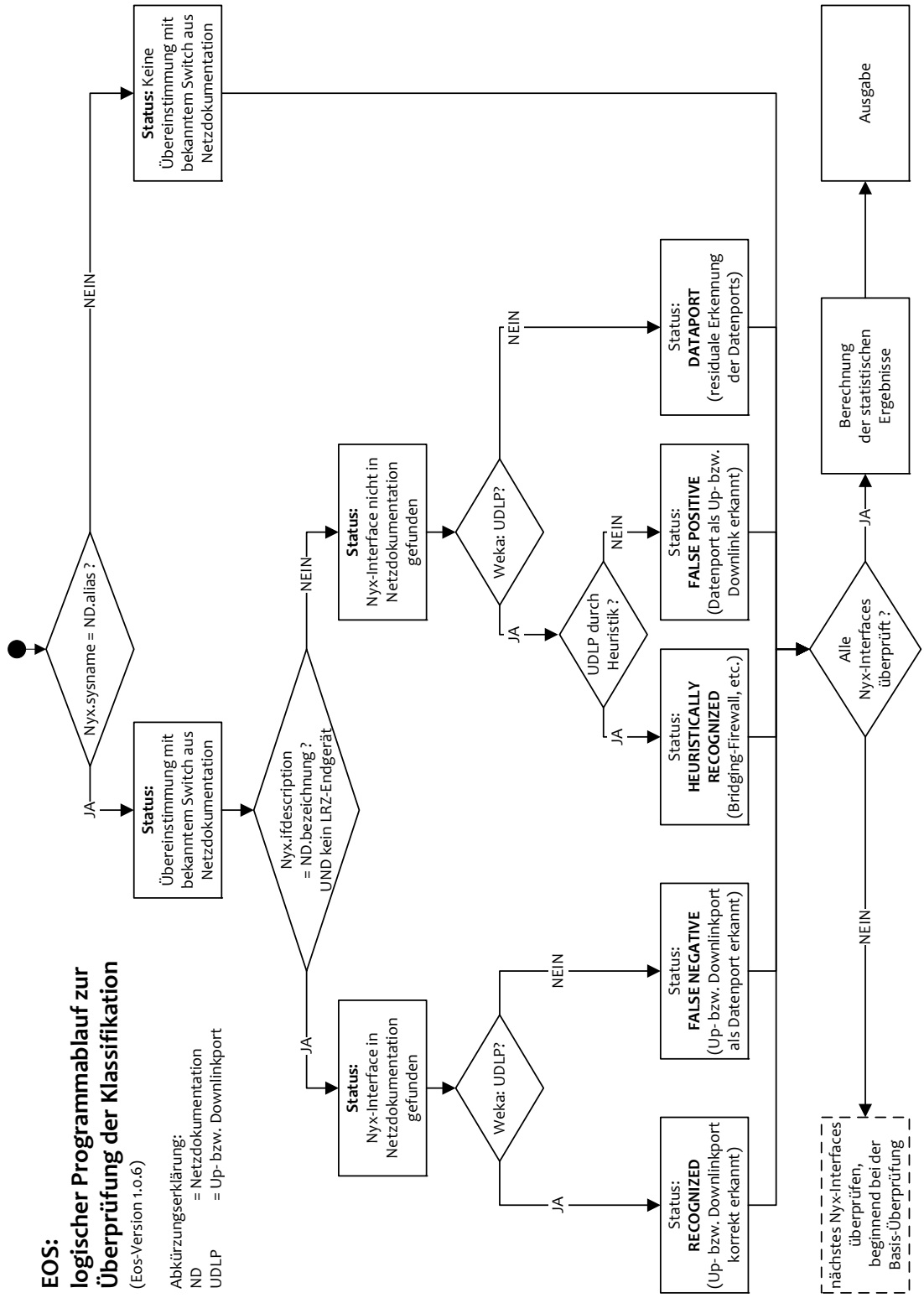


Abbildung 3.3: Eos: Logischer Programmablauf zur Überprüfung der Klassifikation

Im Folgenden werden die prozentualen Werte für die statistisch relevanten Kennzahlen und Verhältnisse ermittelt. Dies geschieht exakt nach der jeweiligen Definition aus Abschnitt 3.1. Dabei werden Sonderfälle separat behandelt, um Divisionen durch Null zu verhindern.

Listing 3.4: Statistische Berechnungen durch Eos

```
# aV
my $adjustmentValue = int ( $nyx_dataport *
                            (1 -$dataport_accuracy_loading));
# aNTN
$nyx_dataport = $nyx_dataport - $adjustmentValue;

my $samples_considered = int( $nyx_updownlinkcount
                            + $nyx_false_negative + $nyx_false_positive
                            + $nyx_dataport + $adjustmentValue);

my $samples_valid = $samples_considered + $heuristically_recognized_ports
                  + $not_found_in_nd;
my $samples_invalid = $samples - $samples_valid;

# false negative and false positive
if ($nyx_false_negative + $nyx_updownlinkcount ne 0) {
  $false_negative_percent = $nyx_false_negative /
                            ($nyx_false_negative + $nyx_updownlinkcount);
}

if (($nyx_false_positive + $nyx_dataport) ne 0) {
  $false_positive_percent = $nyx_false_positive /
                            ($nyx_false_positive + $nyx_dataport);
}

# true negative (sensibility) and true positive (recall or sensitivity)
if (($nyx_updownlinkcount + $nyx_false_negative) ne 0) {
  $true_positive_percent = $nyx_updownlinkcount /
                           ($nyx_updownlinkcount + $nyx_false_negative);
} elsif ($nyx_updownlinkcount eq 0 and $nyx_false_negative eq 0) {
  $true_positive_percent = 1;
}

if (($nyx_false_positive + $nyx_dataport + $adjustmentValue) ne 0) {
  $true_negative_percent = $nyx_dataport / ($nyx_false_positive
                           + $nyx_dataport + $adjustmentValue);
} elsif ($nyx_false_positive eq 0 and $nyx_dataport eq 0 and
         $adjustmentValue eq 0) {
  $true_negative_percent = 1;
}

# adjustment
if (($nyx_false_positive + $nyx_dataport + $adjustmentValue) ne 0) {
  $adjustmentError = $adjustmentValue / ($nyx_false_positive
                   + $nyx_dataport + $adjustmentValue);
} elsif ($nyx_false_positive eq 0 and $nyx_dataport eq 0 and
         $adjustmentValue eq 0) {
  $adjustmentError = 0;
}
```

```
# accuracy
my $accuracy_with_dataports = ($nyx_updownlinkcount + $nyx_dataport) /
                              $samples_considered;

my $accuracy_without_dataports = $true_positive_percent;
```

Anschließend an die Berechnung der prozentualen Verhältnisse – insbesondere der Korrektheit (accuracy) im letzten Teil – prüft Eos deren statistische Qualität, bevor es das Ergebnis ausgibt. Dazu dient der folgende Code-Ausschnitt, dessen mathematische Grundlage im nächsten Abschnitt beschrieben wird. Eos vergleicht im Rahmen dieser Qualitätsprüfung die Zahl der gültigen Ports mit der Zahl der Ports, die nötig sind, damit die Ergebnisse eine gewisse statistische Sicherheit $(1-\alpha)$ haben. Liegt die Zahl der überprüften (und gültigen) Ports über der Zahl der statistisch notwendigen, so sind die Ergebnisse lediglich einer Wahrscheinlichkeit von α inkorrekt.

Listing 3.5: Statistische Qualität des Ergebnisses

```
my $c_N = $num_sysnames;
my $c_alpha1 = 0.05;  # alpha1
my $c_alpha2 = 0.01;  # alpha2
my $c_t1 = 1.96;      # z-quantile: z_(1-(c_alpha1/2)) = z_(0.9725)
my $c_t2 = 2.58;      # z-quantile: z_(1-(c_alpha2/2)) = z_(0.9950)
my $c_p = 0.5;        # maximum
my $c_n1 = ($c_t1*$c_t1  * $c_N * $c_p * (1-$c_p)) /
      ($c_t1*$c_t1 * $c_p * (1-$c_p) + $c_alpha1*$c_alpha1 * ($c_N - 1));
my $c_n2 = ($c_t2*$c_t2  * $c_N * $c_p * (1-$c_p)) /
      ($c_t2*$c_t2 * $c_p * (1-$c_p) + $c_alpha2*$c_alpha2 * ($c_N - 1));
my $correct1 = 1-$c_alpha1;
my $correct2 = 1-$c_alpha2;
```

3.3.3 Statistische Qualität der Resultate

Zur Beurteilung der Aussagekraft der Ergebnisse, die das Skript Eos ermittelt, werden Mittel der Statistik herangezogen. In diesem Zusammenhang sind die Begriffe „Grundgesamtheit“ und „Stichprobe“ zu definieren (in Anlehnung an [11, Abschnitt 6.1: Die Stichprobe]):

Grundgesamtheit: Die Grundgesamtheit N ist die Gesamtmenge an Fällen oder Ereignissen, auf die sich die Aussagen der Untersuchung beziehen soll (vgl. [8, Seite 190]).

Stichprobe: Die Stichprobe n (engl. „sample“) ist eine Teilmenge der Grundgesamtheit, sodass sich die Werte (Mittelwert, etc.) der relevanten Merkmale (Variablen oder Attribute) in der Stichprobe möglichst wenig von der Grundgesamtheit unterscheiden (vgl. [1, Seite 313]). Eine Stichprobe, die alle Elemente der Grundgesamtheit umfasst, bezeichnet man auch als „Vollerhebung“.

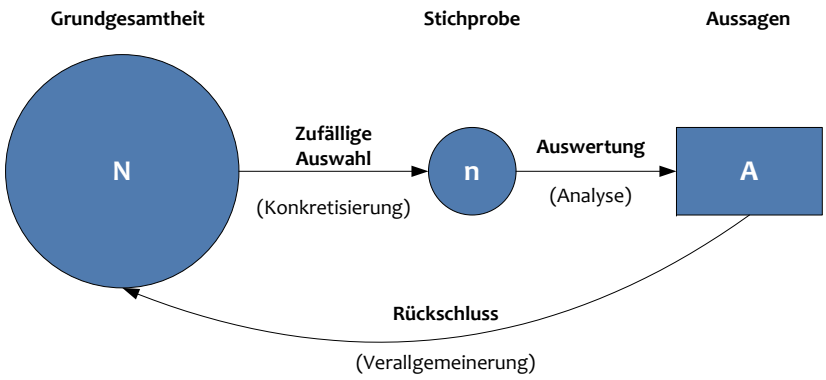


Abbildung 3.4: Zusammenhang von Grundgesamtheit und Stichprobe

An dieser Stelle sei erwähnt, dass eine Vollerhebung offensichtlich bessere Ergebnisse liefert, allerdings genügt meist eine kleinere Menge, eine Stichprobe, um statistisch akzeptable Ergebnisse zu erhalten, die Rückschlüsse auf die Grundgesamtheit erlauben. Damit diese Rückschlüsse akzeptabel sind, muss die Stichprobe natürlich nahezu identische Merkmale haben wie die Grundgesamtheit. Sie muss also ein „verkleinertes Abbild der Grundgesamtheit“ (vgl. [8, Seite 197]) darstellen. Ist dies der Fall, bezeichnet man die Stichprobe als „repräsentativ“.

Beispiel 3.3 (Infratest dimap stellt „Sonntagsfrage“) *Die sogenannte „Sonntagsfrage“ bezeichnet in der Meinungsforschung eine Umfrage zur Ermittlung einer hypothetischen Wahlentscheidung einer Legislaturperiode. Dazu stellt Infratest dimap, aber auch andere Forschungsinstitute, meist etwa 1000 Bundesbürgern die Frage: „Welche Partei würden Sie wählen, wenn am kommenden Sonntag Bundestagswahl wäre?“. Diese Zahl ist ceteris paribus stets nahezu identisch, weil damit eine gewisse Repräsentativität erzielt wird. Es ist nicht nötig (und finanziell sowie organisatorisch auch nicht möglich), alle in Deutschland Wahlberechtigten zu befragen. Infratest dimap nutzt also Stichproben zur Ermittlung repräsentativer Aussagen über die Grundgesamtheit.*

Repräsentativität wird meist durch die Stichprobengröße und das Auswahlverfahren erzeugt. Man unterscheidet hier Zufallsauswahlverfahren, systematische Stichprobenauswahlverfahren und willkürliche Auswahlverfahren. Im Rahmen dieser Arbeit wird ein Zufallsauswahlverfahren aus Basis des SQL-Befehls „order by rand()“ verwendet. Bei diesem Zufallsauswahlverfahren hat jedes Element der Grundgesamtheit die gleiche Wahrscheinlichkeit, in die Stichprobe zu gelangen (deshalb engl. „simple random sampling“). Dafür muss allerdings zuvor eine Liste der Grundgesamtheit (hier in Form einer Datenbanktabelle) vorliegen. Die Wahrscheinlichkeit p (engl. „propability“) eines einzelnen Samples s (gemeint ist hier ein Element der Stichprobe) bei einer nicht-leeren Grundgesamtheit N berechnet sich als

$$p(s) = \frac{1}{N} \quad (0 < p \leq 1)$$

und ist damit offensichtlich stets echt größer Null. Durch die Verwendung von Zufallsauswahlverfahren ist gewährleistet, dass die Methoden der induktiven Statistik anwendbar sind, da bei zufälliger Wahl keine systematischen oder willkürlichen Verzerrungen auftreten sollten.

Die Repräsentativität alleine bestimmt aber noch nicht die Qualität einer Stichprobe. Denn es ist offensichtlich, dass eine Vollerhebung repräsentativ ist. Es wurde aber schon festgestellt, dass eine Vollerhebung oft aus finanziellen oder organisatorisch-logistischen Umständen nicht durchführbar ist. Eine Stichprobe in Form einer Vollerhebung ist demnach zwar in Bezug auf das Ergebnis qualitativ am Besten, aber nicht unter Kosten-Nutzen-Aspekten. Es muss also eine Stichprobengröße ermittelt werden, die sowohl repräsentativ ist als auch möglichst wenig Elemente beinhaltet, um den Nutzen pro Element zu erhöhen. Die Qualität einer Stichprobe misst sich also daran, wie genau der wahre Wert einer Vollerhebung getroffen wird (vgl. [4, Seite 65]). Dabei erlaubt die zufällige Auswahl der Stichprobenelemente die Berechnung eines sogenannten Konfidenzintervalls (Vertrauensbereich).

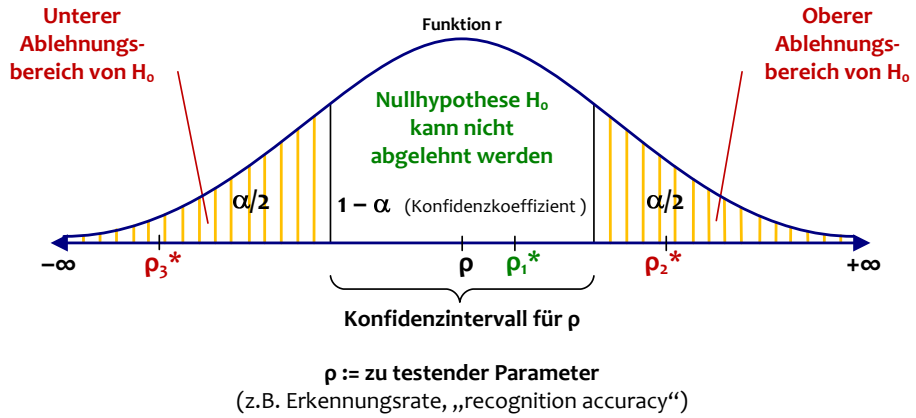


Abbildung 3.5: Konfidenzintervall und Irrtumswahrscheinlichkeit für einen Parameter ρ

Das Konfidenzintervall ist ein gewisser Bereich, den man als Unschärfe in Bezug auf die Präzision der Lageschätzung eines Wertes (genauer eines Parameters, wie z.B. des Mittelwert) interpretieren kann. Für den Parameter kann man anhand einer zuvor festgelegten Wahrscheinlichkeit t ($= 1-\alpha$, vgl. Abschnitt 3.1) sagen, dass er mit eben dieser Wahrscheinlichkeit t innerhalb dieses Bereichs liegt. Ein Vorteil des Konfidenzintervalls gegenüber einer sogenannten Punktschätzung eines Parameters ist die Möglichkeit, das Signifikanzniveau $\alpha = 1-q$ ablesen zu können.

Der Fehler, der durch die Beschränkung der Stichprobe auf n Elemente (mit $n < N$) entsteht und durch die Irrtumswahrscheinlichkeit α in gewisser Weise beschrieben wird, ist also unvermeidbar. Ziel muss es also sein, diesen Fehler zu minimieren und dabei trotzdem die Größe der Stichprobe möglichst gering zu halten. In der Praxis ist es nun so, dass entschieden wird, welcher Stichprobenfehler zulässig ist (z.B. $\pm$ 5 %) und mit welcher Sicherheit (Wahrscheinlichkeit) die Aussage getroffen werden soll [11].

Mit diesem Wissen kann die Stichprobengröße folgendermaßen bestimmt werden:

$$n = \frac{t^2 * N * p(1-p)}{t^2 * p(1-p) * \alpha^2 * (N-1)}$$

wobei
- n : Stichprobengröße
- N : Umfang der Grundgesamtheit
- t : Sicherheitsfaktor (abhängig von der Irrtumswahrscheinlichkeit der Aussagen)
- p : Anteil der Elemente in der Stichprobe, die die Merkmalsausprägung aufweisen (Maximum bei 0,5)
- α : Stichprobenfehler (z.B. bei einer Genauigkeit der Aussage von $\pm$ 5 % ist $\alpha = 0,05$)

Bei der oben angegebenen und in der Praxis häufig verwendeten Irrtumswahrscheinlichkeit von $\alpha = 5\%$ (verteilt auf beide Seiten) und folglich einer Sicherheit von 0.95 % ergibt sich für t ein Wert von ca. 2 (exakt: 1.96). Dieser Wert berechnet sich als Quantil $z_{1-\alpha/2} = z_{0.975} = 1.96$ der Standard-Normalverteilung $N(0,1)$ mit der Verteilungsfunktion

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} * \int_{-\infty}^{z} e^{-z^2/2} \mathrm{d}z$$

Im Falle einer Irrtumswahrscheinlichkeit von $\alpha = 1\%$ (verteilt auf beide Seiten) und damit einer Sicherheit von 0.99 % ist $t = 2.5758 \approx 2.58$.

Vereinfacht man die Formel mit diesen Werten für t und setzt man dabei für p das Maximum von 0.5 ein, da die genaue Wahrscheinlichkeit von p meist im Vorhinein nicht bekannt ist, erhält man folgende Formel für die Größe der Stichprobe:

$$n = \frac{N}{1 + \alpha^2 * (N-1)},$$

denn der quadrierte Sicherheitsfaktor sowie der Faktor $p(1-p)$ können ohnehin partiell gekürzt werden. Als grober Richtwert kann man weiter vereinfachend folgende Formel nutzen:

$$n = \frac{1}{\alpha^2} \tag{3.15}$$

Diese stellt eine „obere Grenze“ dar, wie man in der folgenden Tabelle 3.8 sieht. Außerdem kann man erkennen, dass für das Szenario MWN etwa 9000 Samples als Stichprobe benötigt werden. Auffällig ist dabei, dass der Umfang der Stichprobe bei einer genügend großen Grundgesamtheit unabhängig von dieser ist. Dies wird besonders bei der Irrtumswahrscheinlichkeit von 5 % und der maximalen Stichprobengröße von 400 Elementen deutlich.

N	**Fehler** $\pm 5\%(\alpha = 0.05)$		**Fehler** $\pm 1\%(\alpha = 0.01)$	
	$n = \frac{N}{1 + \alpha^2 * (N-1)}$	$n = \frac{1}{\alpha^2}$	$n = \frac{N}{1 + \alpha^2 * (N-1)}$	$n = \frac{1}{\alpha^2}$
100	80	400	99	10.000
500	222	400	476	10.000
1.000	286	400	909	10.000
5.000	370	400	3334	10.000
10.000	385	400	5000	10.000
20.000	392	400	6667	10.000
65.000	397	400	8667	10.000

Tabelle 3.8: Stichprobengröße bei unterschiedlichem Umfang der Grundgesamtheit und einer Irrtumswahrscheinlichkeit α von 5 bzw. 1 % (in Anlehnung an [11])

3.4 Ist-Analyse

In diesem Abschnitt soll nun schließlich das Skript Eos, das im vorherigen Kapitel 3.3 ausführlich beschrieben wurde, dazu verwendet werden, das Lokalisierungssystem Nyx im Ist-Zustand zu beurteilen und die Erkennungsleistung zu quantifizieren. Das geschieht im Rahmen dieser Arbeit auf Basis der Arbeits- bzw. Nullhypothese, wie sie im Abschnitt 3.1 definiert wurde. In der Analyse wird demnach prinzipiell versucht, die Arbeitshypothese (annähernd) zu bestätigen oder sie zu widerlegen. Zweiteres geschieht, indem eine neue (aber exaktere) Hypothese als Ergebnis der Analyse durch Eos erstellt wird.

Die Analyse selbst geschieht in zwei Stufen: Zuerst beschränkt sich die Analyse auf einen vordefinierten Bereich, der besonders gut zu kontrollieren ist, sodass die Ergebnisse exakt überprüfbar sind. Im Rahmen dieser Arbeit stellt der LRZ Rechnerwürfel diesen Bereich dar. Innerhalb dieses Bereichs wird eine gewisse Anzahl von Ports zufällig ausgewählt und deren Klassifikation dann durch eine Sichtung manuell überprüft. Anschließend wird das Ergebnis mit dem Resultat von Eos verglichen. Da beide Stichproben einen Umfang von mindestens 100 Elementen haben und ein Zufallsauswahlverfahren auf Basis des SQL-Befehls „order by rand()“ verwendet wird, kann man davon ausgehen, dass beide Ergebnisse ungefähr mit einer statistischen Wahrscheinlichkeit von 90% ($1 - \alpha = 1 - 0,1 = 0,9$) korrekt sind (vgl. Richtwert 3.3.3), Eos sogar zu 95%, da leicht etwa 1000 Ports automatisiert überprüft werden können. Für eine höhere statistische Sicherheit müssten an dieser Stelle wesentlich mehr Ports manuell durch Sichtung überprüft werden (können), was im Rahmen dieser Arbeit und im Allgemeinen praktisch kaum durchführbar ist.

Aus diesem Grund wird im zweiten Schritt der Analysebereich auf Kosten der manuellen Überprüfbarkeit erweitert, um dadurch die statistische Sicherheit zu erhöhen. Diese lässt sich so auf über 95% bzw. bei mehr als 10.000 gültigen Ports sogar auf über 99% steigern (vgl. Abschnitt 3.3.3). Auf diese Weise ist es möglich, die Arbeitshypothese zu bestätigen oder zu

widerlegen. Da bei einer steigenden Anzahl an prüfbaren Ports die Bedeutung der Datenports, die im Vergleich einen wesentlich höheren Anteil im Szenario ausmachen, aber stark zunimmt, muss auf die Wahl des Gewichtungsfaktors geachtet werden. Im Rahmen dieser Arbeit kann dies aber größtenteils vernachlässigt werden, weil vor allem die True Positive Rate (TP) für die Einsatzzwecke von Nyx, die im Szenario angedeutet wurden, und die Filterung für die Datenbank entscheidend sind. Die absolute Richtigkeit (accuracy) ist für den Anwendungsfall nur implizit von Bedeutung, da das vorwiegende Ziel die Erkennung von Inter-Switch-Verbindungen anhand von Up- bzw- Downlinkports zur Filterung ist.

Anschließend an diese Ist-Analyse folgt eine Untersuchung der Gründe für Klassifikationsfehler. Das nächste Kapitel beinhaltet dann die Beschreibung der Verbesserungsmöglichkeiten, die sich aus den Fehlern heraus ergeben.

3.4.1 Ist-Zustand mit Beschränkung auf den LRZ Rechnerwürfel

Für die Analyse des LRZ Rechnerwürfels als kleiner, kontrollierter und für die manuelle Sichtung geeigneter Teilbereich des MWNs muss zunächst eine geeignete Stichprobe für die manuelle Sichtung erzeugt werden. Die Auswahl der zu überprüfenden Ports muss dabei auf den Rechnerwürfel, also den Unterknotenbezirk „WR“ des Münchner Wissenschaftsnetzes, beschränkt werden. Außerdem sollen keine Router (Netzkomponenten mit „csr“ im sysname), keine Geräte ohne Kennung (sysname) und keine ungültigen Portbeschreibungen enthalten sein. Letzteres wird in SQL durch sogenannte Wildcards (Platzhalter) realisiert: An der Stelle des Zeichens „_“ kann genau ein Zeichen stehen, sodass „__“ bzw. „___“ im Normalfall gültige Portbeschreibungen erzeugen und eine ausreichende Kontrollbedingung für die Anforderung darstellen. Das komplette SQL-Statement ist in Listing 3.6 aufgeführt.

Listing 3.6: Zufällige Auswahl von Ports im LRZ Rechnerwürfel

```
SELECT device, ifindex, ifdescription, ifspeed,
       updownlink, sysname, syslocation
FROM   interface, device
WHERE  interface.device = device.id
   AND device.sysname LIKE '%WR%'
   AND device.sysname NOT LIKE '%csr%'
   AND device.syslocation NOT LIKE ''
   AND ( interface.ifdescription LIKE '___'
      OR interface.ifdescription LIKE '__' )
ORDER BY RAND()
LIMIT 100;
```

Die aus diesem SQL-Statement resultierende Stichprobe ist im Anhang D aufgeführt. Das Ergebnis der Überprüfung ist eine Erkennungsrate (accuracy) von 99%. Führt man zusätzlich eine Analyse mit Eos aus, erhält man einen ähnlich guten Wert von etwa 94%, ohne Datenports sogar ebenfalls 99% (siehe auch Tabelle 3.9).

```
EOS results:
--------------------------------------------------------------------------------
Maximum number of samples tested                        : 20000
|
|- Invalid switches (or residual value to maximum)      : 14171
|
|- Valid switches (sysname not empty, ifdescription < 3): 5829
   |
   |- Switches not found in NETZDOKU (not considered)   : 308
   |
   |- Heuristically recognized ports (not considered)   : 611
   |
   |- Number of considered samples                      : 4910
      |
      |- Number of Updownlinkports                      : 367
         |- True positive ratio: TP = 99.19%
      |
      |- Number of FN: Updownlink classified as dataport: 3
         |- False negatives ratio: FN = 0.81%
      |
      |- Adjusted Number of Dataports                   : 4288
         |- True negatives ratio: TN = 94.45%
      |
      |- Number of FP: Dataport classified as updownlink: 165
         |- False positive ratio: FP = 3.71%
      |
      |- Adjustment value                               : 87
         |- Adjustment Error: AE = 1.92%
      |
      |- Accuracy  (no dataports included)              : 99.19%
      |- Accuracy  (dataports included)                 : 94.81%
--------------------------------------------------------------------------------

(Results are correct with 95.00 percent confidence.
Number of valid samples must be higher than 13841  to get results with
99.00 percent correctness.)
```

Abbildung 3.6: Ausgabe von Eos für den LRZ Rechnerwürfel

Da diese Werte allerdings nur eine statistische Sicherheit von 95% in Bezug auf das Gesamtnetz haben, die Stichprobe also nicht ausreichend groß für eine repräsentative Aussage über das Gesamtnetz ist, kann an dieser Stelle lediglich festgehalten werden, dass die Arbeitshypothese von 80 bis 85 Prozent für den LRZ Rechnerwürfel widerlegt ist. Ob sich dieses gute Ergebnis aber auch auf das Gesamtnetz übertragen lässt, wird nachfolgend überprüft.

Durchlauf	1	2	3	4	5	Mittelwert
NTP	367	357	366	358	367	**363**
TP	99,19%	99,71%	99,19%	99,17%	99,19%	**99,29%**
NFN	3	3	3	3	3	**3**
FN	0,81%	0,83%	0,81%	0,83%	0,81%	**0,82%**
NTN	4288	4298	4308	4297	4294	**4297**
TN	94,45%	94,42	94,49%	94,44%	94,50%	**94,46%**
NFP	165	167	164	166	163	**165**
FP	3,63%	3,67%	3,60%	3,65%	3,59%	**3,63%**
aV	87	87	87	87	87	**87**
aE	1,92%	1,91%	1,91%	1,91%	1,91%	**1,91%**
Accuracy*	99,19%	99,17%	99,19%	99,17%	99,19%	**99,18%**
Accuracy**	94,18%	94,77%	94,85%	94,79%	94,85%	**94,81%**

(* ohne Datenports, ** dataport_loading=0,98)

Tabelle 3.9: Zusammenfassung der Ergebnisse von Eos für die Überprüfung des LRZ Rechnerwürfels

3.4.2 Ist-Zustand des gesamten Münchner Wissenschaftsnetzes

Zur Ermittlung des Ist-Zustands für das Münchner Wissenschaftsnetz kann im Gegensatz zur Überprüfung des LRZ Rechenwürfels keine manuelle Sichtung durchgeführt werden, weil sich das MWN über sehr viele Standorte erstreckt (vgl. Tabelle 1.1) und die Grundgesamtheit aller Ports für eine manuelle Stichprobenerhebung zu groß ist. Aus diesem Grund beschränkt sich die Analyse auf die Auswertung der Ergebnisse, die Eos unter Verwendung der LRZ Netzdokumentation liefert.

Das Skript liefert mit einer statistischen Sicherheit von 99% eine Erkennungsrate von 94,89% ohne Berücksichtigung der Datenports und eine Rate von 95,57% unter Einbeziehung der Datenports. Mehrere Durchläufe, die in Tabelle 3.10 zu sehen sind, erhöhen diese Werte im Mittel nur um maximal etwa einen Prozentpunkt, was im Bereich der statistischen Konfidenz liegt. In jedem Fall bleibt die Erkennungsrate (accuracy) aber unterhalb der eingestellten Erkennungsrate der Datenports (dataport_loading) von 98%. Aus diesem Grund sollte eine Anpassung dieser Gewichtung in Erwägung gezogen werden, um adäquate Ergebnisse zu erhalten. Dieser Punkt wird aber an dieser Stelle in der Praxis nicht weiter analysiert, sondern lediglich die entsprechenden mathematischen Werte nach einer Anpassung in Tabelle 3.11 angegeben. Man stellt fest, dass die Erkennungsleistung für eine geringere Gewichtung der Datenports deutlich absinkt. Bei einer Gewichtung der Datenports, die mit 0,95 gleichwertig zur Erkennung der Up- bzw. Downlinkports wäre, ergibt sich lediglich eine Erkennungsrate von insgesamt ca. 92 Prozent.

Durchlauf	1	2	3	4	5	Mittelwert
NTP	334	372	339	310	355	**342**
TP	94,89%	96,88%	95,22%	95,68%	95,95%	**95,72%**
NFN	18	12	17	14	15	**15**
FN	5,11%	3,12%	4,78%	4,32%	4,05%	**4,28%**
NTN	14321	14357	14317	14399	14289	**14337**
TN	95,58%	95,45%	95,31%	95,23%	95,26%	**95,37%**
NFP	370	392	412	428	420	**404**
FP	2,47%	2,61%	2,74%	2,83%	2,80%	**2,69%**
aV	292	293	292	293	291	**292**
aE	1,95%	1,95%	1,94%	1,94%	1,94%	**1,94%**
Accuracy*	94,89%	96,88%	95,22%	95,68%	95,95%	**95,72%**
Accuracy**	95,57%	95,48%	95,31%	95,24%	95,28%	**95,38%**

(* ohne Datenports, ** dataport_loading=0,98)

Tabelle 3.10: Zusammenfassung der Ergebnisse von Eos für die Überprüfung des Münchner Wissenschaftsnetzes (MWN)

dataport_loading	0.98	0,97	0.96	0.95	0.90
TP	94,89%	94,89%	94,89%	94,89%	94,89%
FN	5,11%	5,11%	5,11%	5,11%	5,11%
NTN	**14321**	**14175**	**14029**	**13883**	**13152**
TN	**95,58%**	**94,61%**	**93,63%**	**92,66%**	**87,78%**
FP	2,47%	2,47%	2,47%	2,47%	2,47%
aV	**292**	**438**	**584**	**730**	**1461**
aE	**1,95%**	**2,92%**	**3,90%**	**4,87%**	**9,75%**
Accuracy*	**94,89%**	**94,89%**	**94,89%**	**94,89%**	**94,89%**
Accuracy	**95,57%**	**94,61%**	**93,66%**	**92,71%**	**87,94%**

(* ohne Datenports)

Tabelle 3.11: Ergebnisse von Eos für das Münchner Wissenschaftsnetz (MWN) in Abhängigkeit von der Gewichtung der Korrektheit der Datenporterkennung

3.5 Analyse von Nyx

Nach der Ermittlung des Ist-Zustands folgt in diesem Abschnitt eine genaue Analyse von Nyx zur anschließenden Verbesserung des Systems, soweit dies überhaupt möglich ist und Fehler entdeckt werden. Dazu wird im Wesentlichen die Modellausprägung des Entscheidungsbaums auf Basis des Klassifikationsalgorithmus J4.8 diskutiert. Für die in Kapitel 2.2.3 beschriebenen Kriterien, die vor dieser Arbeit alle Verwendung fanden, ergibt sich für die Trainingsdaten folgender Entscheidungsbaum:

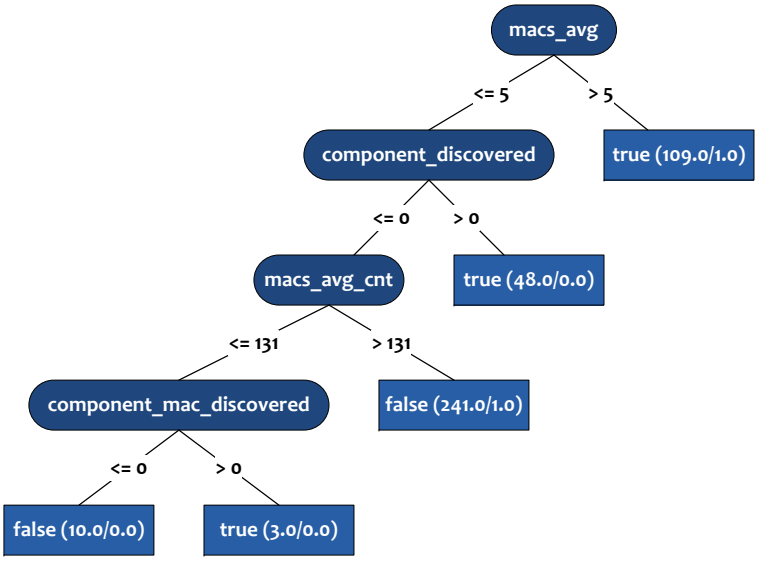


Abbildung 3.7: Entscheidungsbaum durch J4.8 für Nyx vor dieser Arbeit

Betrachtet man diesen Entscheidungsbaum genauer, stellt man fest, dass dieser offensichtlich Fehler aufweist. Diese Fehler sind nicht algorithmischer, sondern inhaltlicher Natur. Der Klassifikationsalgorithmus erstellt zwar den Entscheidungsbaum korrekt nach dem in Kapitel 2.2.4.1 beschriebenen Verfahren, die bereitgestellten Trainingsdaten beinhalten aber offenbar irrelevante Daten. So sollte die Zahl, wie oft die durchschnittliche Anzahl der registrierten MAC-Adressen bestimmt wurde („macs_avg_cnt“), nicht im Entscheidungsbaum auftauchen. Dieser und weitere Fehler werden im Folgenden genauer diskutiert.

3.5.1 Fehler im Klassifikationsmodell

Wie bereits beschrieben, beinhaltet der Entscheidungsbaum das Attribut „macs_avg_cnt“. Dieses ist aber lediglich ein Zähler, wie schon die Bezeichnung vermuten lässt. Dieser beinhaltet keine relevante Information über die Art des Ports. Es gibt also keine logische Verbindung zwischen dem Stand dieses Zählers und der Tatsache, ob es sich bei einem Port um einen Up- bzw. Downlinkport oder einen Datenport handelt. Der Zähler gibt lediglich an, wie oft die durchschnittliche Anzahl an registrierten MAC-Adressen für den betrachteten Port ermittelt wurde. Aus diesem Grund ist dieses Attribut aus dem Baum zu entfernen. Dadurch allein verbessert sich bereits die Qualität der Klassifikation.

Zur Beurteilung der Auswirkung dieser Korrektur wird an dieser Stelle zuerst eine sogenannte „stratified 10-fold cross-validation“ (stratifizierte 10-fache Kreuzvalidierung) als Testverfahren verwendet. Dazu wird das Trainingsset, das im Allgemeinen aus N Instanzen bestehen soll, zunächst in k Teilmengen $T_1, T_2, \ldots, T_k$ (mit $k \leq N$, hier: $k = 10$) zerlegt. Die stratifizierte 10-fache Kreuzvalidierung achtet an dieser Stelle im Gegensatz zur einfachen Kreuzvalidierung darauf, dass jede der k Teilmengen eine annähernd gleiche Verteilung besitzt, und verringert so die Varianz der Abschätzung. Nach der Partitionierung werden k Testdurchläufe ausgeführt, bei denen die jeweils i-te Teilmenge T_i als Testmenge und die verbleibenden $k-1$ Teilmengen $\{T_1, T_2, \ldots, T_k\} \backslash \{T_i\}$ als Trainingsmengen verwendet werden. Der Durchschnitt aus den Einzelfehlerquoten der k Einzeldurchläufe ergibt dann anschließend die Gesamtfehlerquote.

Das Ergebnis der Kreuzvalidierung für die Trainingsmenge vor der Korrektur lässt sich unter anderem durch die folgenden Maße für numerische Vorhersagen beurteilen, wobei p die vorhergesagten (engl. „predicted“) und a die tatsächlichen (engl. „actual“) Werte darstellen und n die Zahl der Fälle ist (aus: [20, Seite 178]):

Bezeichnung (engl.)	**Formel**
mean-squared error	$\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}$
root mean-squared error	$\sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}}$
relative squared error	$\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{(a_1 - \overline{a})^2 + \ldots + (a_n - \overline{a})^2}$, wobei $\overline{a} = \frac{1}{n}\sum_i a_i$
root relative squared error	$\sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{(a_1 - \overline{a})^2 + \ldots + (a_n - \overline{a})^2}}$, wobei $\overline{a} = \frac{1}{n}\sum_i a_i$

Tabelle 3.12: Maße zur Bestimmung der Qualität einer numerischen Vorhersage

Durch die Änderung an den Trainingsdaten sinkt offensichtlich der gewurzelte mittlere quadrierte Fehler von etwa 0,0854 auf 0,0697. Deutlicher wird die Verbesserung durch Betrachtung des gewurzelten relativen quadrierten Fehlers, der von ca. 17,52 auf 14,29 Prozent sinkt. Diese Verbesserung wird nochmals in Tabelle 3.13 gegenübergestellt. Wie signifikant sie tatsächlich ist, wird sich am Ende in der Analyse durch Eos zeigen.

Führt man nun im zweiten Schritt eine Berechnung des Gain-Ratio (vgl. Kapitel 2.2.4.2) durch, so bestätigt sich das Ergebnis aus dem vorherigen Schritt: Das Attribut „mac_avg_cnt“ nimmt mit einem Wert von ca. 0,09 den letzten Platz in der Rangfolge der Attribute in Bezug auf den Gain ein. Anhand dieser zweiten Auswertung bestätigt sich das vorherige Ergebnis. Der Informationszugewinn dieses Attributs ist nahezu Null. Es liegt also nahe, dieses Attribut aus der Trainingsmenge zu entfernen, zumal es sicherlich inhaltlich irrelevant ist. Eine verbesserte Wahl der Attribute und der Trainingsdaten wird deshalb im folgenden Kapitel 4 genauer beschrieben.

Vor Entfernung des Kriteriums „macs_avg_cnt“:		
Bezeichnung	**Werte**	
Korrekt klassifizierte Fälle	408	99,2701 %
Falsch klassifizierte Fälle	3	0,7299 %
root mean squared error		0,0854
root relative squared error		17,5165 %

Nach der Entfernung des Kriteriums „macs_avg_cnt“:		
Bezeichnung	**Werte**	
Korrekt klassifizierte Fälle	409	99,5134 %
Falsch klassifizierte Fälle	2	0,4866 %
root mean squared error		0,0697
root relative squared error		14,2946 %

Tabelle 3.13: 10-fache stratifizierte Kreuzvalidierung mit und ohne Berücksichtigung des Kriteriums „macs_avg_cnt“

Bewertung:	Gain Ratio feature evaluator	
Suche:	Attribute ranking	
Ranking	**Gain-Ratio**	**Attributname**
1	0,5482	macs_avg
2	0,505	component_discovered
3	0,3509	traffic_percent_avg
4	0,3184	tagged
5	0,2498	custom_criteria
6	0,214	component_mac_discovered
7	0,0902	macs_avg_cnt

Tabelle 3.14: Analyse des Gain-Ratio der Attribute vor dieser Arbeit

3.5.2 Sonstige Erkennungsfehler

Zusätzlich zum inkorrekten Klassifikationsmodell wurden im Rahmen dieser Arbeit Klassifikationsfehler festgestellt, die nicht in direkter Verbindung mit dem Lernalgorithmus und den verwendeten Trainingsdaten stehen. Diese werden in den folgenden Unterabschnitten genauer beschrieben.

3.5.2.1 Reset-Fehler

Bei der manuellen Sichtung im LRZ Rechnerwürfel ist aufgefallen, dass das Attribut „component_mac_discovered“ nicht zurückgesetzt wird. Dies führt zu inkonsistenten und fehlerhaften Zuständen und es gehen unter gewissen Umständen Aktualität und Aussagekraft dieses (unter normalen Umständen) sicheren Kriteriums verloren. Da sich ein solcher Fehler prinzipiell auf alle Attribute übertragen lässt, bilden sie eine Fehlerklasse. Diese Klasse der Reset-Fehler ist im Allgemeinen in der Implementierung begründet, nicht im Klassifikationsalgorithmus bzw. dessen Trainingsdaten. Zur Behebung des Problems muss also der Quellcode von Nyx korrigiert werden. Dies erfolgte abseits dieser Arbeit und wird an dieser Stelle nicht weiter beschrieben. Es kann aber davon ausgegangen werden, dass auch dieser Fehler (für das beschriebene Attribut) korrigiert wurde.

3.5.2.2 LRZ VMware-Cluster

Ein zweiter wesentlicher Grund für fehlerhafte Erkennungen ist die mangelnde Unterscheidbarkeit der Portarten anhand der vorgestellten Kriterien. Dies sei am Beispiel „LRZ VMware-Cluster“ erklärt. Dabei handelt es sich um Server, die sehr viele virtuelle Maschinen beherbergen. Die Zahl der virtuellen Netzwerkkarten ist sehr hoch, sodass auch die Anzahl an registrierten MAC-Adressen im Mittel überdurchschnittlich hoch ist. Außerdem unterstützen unter anderem die Server des VMware-Clusters die Markierung von Netzpaketen zur Zuordung zu einem bestimmten VLAN (VLAN-tagging). Der maschinelle Lernalgorithmus klassifiziert die Ports dieser VMware-Server deshalb immer als Up- bzw. Downlinkports. Tatsächlich sind es jedoch Datenports, weil die Server als Endgeräte zu werten sind. Technisch lässt sich diese Fehlklassifikation aber kaum vermeiden, weil das charakteristische Muster des LRZ VMware-Clusters eher einer Netzkomponente als einem Endgerät entspricht.

3.5.2.3 False-Negatives und Bridging-Firewalls

Ein komplexerer Erkennungsfehler durch Ununterscheidbarkeit entsteht durch False-Negatives in Verbindung mit Bridging-Firewalls. Bei falsch-negativer Klassifikation wird zum einen mindestens ein Up- bzw. Downlinkport fälschlicherweise als Datenport erkannt. Das führt (immer) zu überflüssigen Daten in der Datenbank. Zum anderen kommt im Fall von Bridging-Firewalls hinzu, dass diese mit beiden Ports am selben Switch angeschlossen sind. Allein die Ports gehören zu unterschiedlichen VLANs: dem sogenannten internen respektive externen VLAN (siehe Abbbildung 3.8).

Bei einer solchen Konstellation scheint dann für Nyx die betrachtete MAC-Adresse ständig zwischen zwei Ports (und VLANs) hin und her zu springen. Es wird nämlich in der Nyx-Datenbank für jeden Wechsel des Ports (und damit auch der VLAN-Nummer) am Switch ein neuer Eintrag mit dem aktuellen Zeitstempel erstellt bzw. ein Update der vorhandenen Daten

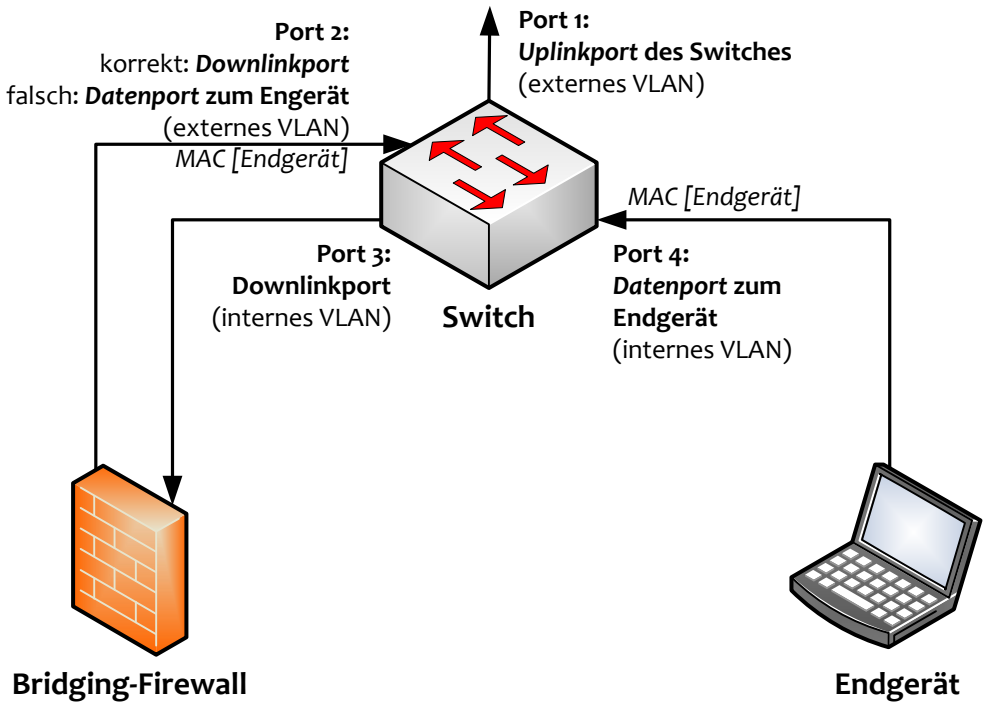


Abbildung 3.8: Bridging-Firewall-Konstellation

durchgeführt. Am Switch selbst ist die MAC-Adresse natürlich stets sowohl am Datenport als auch am Downlinkport zur Bridging-Firewall sichtbar.

Man kann in dieser Situation also nicht mehr sagen, welcher der beiden Ports nun der tatsächliche Datenport ist, weil (mindestens) ein False-Negative vorliegt und beide Ports das gleiche Muster aufweisen. Auch die Verkehrsflussdaten geben hier keine Auskunft. Es ist deshalb eine zusätzliche Information nötig. Das einzige Unterscheidungsmerkmal, das überhaupt als Zusatzinformation zur Verfügung steht, ist das VLAN. Zur Ermittlung des tatsächlichen Datenports muss also der Port mit einer internen VLAN-Nummer bestimmt werden. Da es bei der Zuweisung dieser VLAN-Nummern zwar gewisse Konventionen gibt, aber auch einige Ausnahmen existieren, wird bei Nyx manuell eine Liste der internen VLAN-Nummern verwaltet. So ist es möglich, Bridging-Firewall-Konstellationen durch eine Heuristik zu behandeln: Stellt die Bridging-Firewall-Heuristik eine MAC-Adresse fest, die zwischen zwei Ports hin und her springt, wird anhand der internen VLAN-Nummer der tatsächliche Datenport ermittelt. Der andere (Bridging-Firewall-)Port wird als Up- bzw. Downlinkport behandelt und dementsprechend nicht in der Datenbank gespeichert, sondern verworfen. Lediglich der tatsächliche Datenport wird zur Lokalisierung gespeichert.

Der Vollständigkeit halber sei an dieser Stelle erwähnt, dass eine Aufnahme des Kriteriums „VLAN-Nummer“ in die Trainingsdaten wahrscheinlich keine wesentliche Verbesserung darstellt. Der Grund dafür ist, dass die manuell gepflegte Liste der internen VLAN-Nummern in Kombination mit der aktuell verwendeten Heuristik bereits theoretisch exakte Ergebnisse liefert. Es war jedoch im Rahmen dieser Arbeit nicht mehr möglich, diese Aussage empirisch zu belegen.

Das Beispiel der Bridging-Firewall-Konstellation verdeutlicht somit, dass Sonderfälle existieren, die aktuell durch Mustererkennung alleine nicht sicher erkannt werden können. Gewisse Klassifikationsfehler lassen sich wahrscheinlich sogar nie allein durch Optimierung der vorhandenen Trainingsdaten oder des Lernalgorithmus korrigieren. Da es sich jedoch lediglich um Einzelfälle handelt, kann man im Allgemeinen davon ausgehen, dass sich Verbesserungen an Trainingsdaten und Lernalgorithmus auf die Erkennungsrate auswirken. Dies belegen Änderungen im Rahmen dieser Arbeit, die im folgenden Kapitel beschrieben werden.

4 Verbesserungen

In diesem Kapitel soll nun aufgrund der Erkenntnisse aus den vorherigen Abschnitten die Erkennungsleistung verbessert werden, soweit dies nicht bereits durch Korrekturen der Implementierung von Nyx geschehen ist. Wesentliches Augenmerk wird dabei auf dem Klassifikationsmodell und den Trainingsdaten liegen. Beide korrelieren nämlich sehr stark: Durch Anpassung der Trainingsdaten ändert sich meist ceteris paribus das Klassifikationsmodell. Bei einer Wahl eines gänzlich anderen Klassifikationsalgorithmus ist dies sogar höchst wahrscheinlich der Fall. Auch diese Möglichkeit der Verbesserung wird in diesem Kapitel diskutiert, indem überprüft wird, ob es einen besseren Klassifikationsalgorithmus für das verbesserte Trainingsset gibt.

4.1 Korrektur des Klassifikationsmodells

Wie bereits im vorherigen Kapitel angedeutet, wird an dieser Stelle diskutiert, inwieweit die verwendeten Kriterien sinnvoll sind. In der Analyse wurde bereits festgestellt, dass das Kriterium „macs_avg_cnt“ inhaltlich irrelevant und für den Lernalgorithmus informationstheoretisch unbrauchbar ist. Aus dem Grund wird zunächst dieses Kriterium entfernt und die Auswirkung beobachtet.

Nach Korrektur der Trainingsdaten wird die Aussagekraft des Kriteriums „traffic_percent_avg“ diskutiert. Im Wesentlichen beschäftigt sich der Abschnitt zu diesem Kriterium also mit der Frage, ob der Datenverkehr Aufschluss über die Art des Ports gibt. Je nach Ergebnis ist auch hier eine Korrektur der Trainingsdaten erforderlich. Wie diese genau aussieht, hängt ebenfalls vom Ergebnis ab, beschränkt sich aber offensichtlich auf die beiden Möglichkeiten, das Kriterium entweder zu entfernen oder es in veränderter Form neu zu trainieren.

4.1.1 Entfernung des Kriteriums „macs_avg_cnt“

Entfernt man das Kriterium „macs_avg_cnt“ aus den Trainingsdaten, so sinkt der mittlere quadratische Fehler, wie bereits im vorherigen Kapitel beschrieben, von 72,93 auf 48,58 Prozent und es ergibt sich ein neuer Entscheidungsbaum, der in Abbildung 4.1 zu sehen ist.

Gleichzeitg steigt die relative Klassifikationskorrektheit bei einer 10-fachen Kreuzvalidierung der Trainingsdaten um ca. 33 Prozent:

$$\text{relative Verbesserung} = \frac{99,5134 - 99,2701}{100 - 99,2701} \approx 0,3333 = 33,33\%$$

Grundlage dieser Berechnung ist die Annahme, dass eine Verbesserung im unteren Prozentbereich im Bereich des Data Minings leichter möglich ist als im oberen Bereich. Nach dieser Annahme entspricht eine Verbesserung von 50 auf 55 Prozent (relativ: $\frac{55-50}{100-50} = 10\%$) lediglich einer relativen Verbesserung von 90 auf 91 Prozent (relativ ebenfalls 10%).

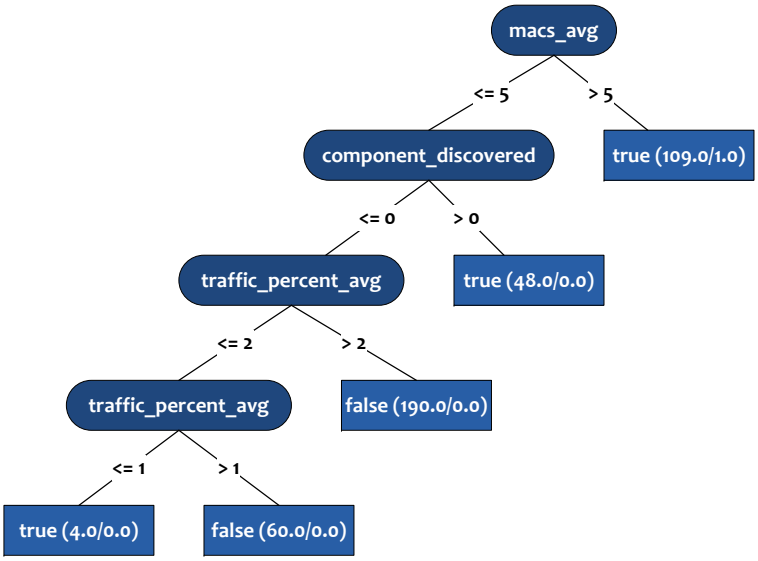


Abbildung 4.1: Neuer Entscheidungsbaum durch J4.8 für Nyx

4.1.2 Entfernung des Kriteriums „traffic_percent_avg“

In diesem Abschnitt soll nun die Aussagekraft des Kriteriums „traffic_percent_avg“ kurz diskutiert werden. Basis dieser Untersuchung sei ohne Beschränkung der Allgemeinheit (o.B.d.A) ein Switch mit genau einem Uplink und mindestens zwei Datenports, an denen Endgeräte angeschlossen sind, gegeben. Der Fall, dass lediglich ein Datenport in Gebrauch ist, wird an dieser Stelle ausgeschlossen, weil zum einen der Switch unnötig und zum anderen die Analyse dieses Falls nicht sinnvoll durchführbar wäre.

Ausgehend von dem beschriebenen Fall legt die Verwendung des Traffic-Kriteriums stets die Annahme zugrunde, dass hoher Datenverkehr charakteristisch für den Uplink ist (vgl. [7, 4.1 Topologieerkennung, Absatz 6]). Der Grund für diese Annahme ist die Vorstellung, dass ein Endgerät hauptsächlich über den Uplink des Switches kommuniziert, sodass durchschnittlich bis zu 50 Prozent des Datenverkehrs am Uplinkport des Switches anfallen, hingegen nur ca. 25 Prozent an den beiden Datenports. Dieser Fall tritt aber nur genau dann ein, wenn die Endgeräte nur unwesentlich untereinander kommunizieren.

Findet eine Kommunikation vermehrt zwischen den Endgeräten (z.B. Kommunikation: Client-Server) statt, was in der Praxis durchaus der Fall sein kann, so gilt die Annahme nicht mehr und das Kriterium „traffic_percent_avg“ verliert seine Aussagekraft. Es besteht nämlich zusätzlich zur Annahme die Möglichkeit, dass die Datenports einen höheren Anteil am Datenverkehr (als der Uplinkport) aufweisen, wenn sie Daten direkt austauschen ohne den Uplink zu nutzen. Es kann aber auch sein, dass die Anteile unter den Datenports und dem Uplinkport gleich verteilt sind. Dies ist der Fall, wenn die Kommunikation ausgewogen über den Uplink und zwischen den Endgeräten erfolgt.

Insgesamt kann man also festhalten, dass es kein eindeutiges Muster gibt und das Kriterium in der aktuellen Form keinen qualitativen Wert hat. Diese Erkenntnis hat zur Folge, dass auch das Kriterium „traffic_percent_avg“ entfernt wird, weil es keine eindeutige Auskunft über die Art eines Ports gibt. Die Auswirkungen dieser Änderung zeigen sich in der 10-fachen

stratifizierten Kreuzvalidierung der Trainingsdaten: Die Wurzel des mittleren quadratischen Fehlers steigt von vormals 0,0697 auf 0,1099 und die Wurzel des relativen quadrierten Fehlers auf ca. 22,55 Prozent (von ursprünglichen 14,29% vor dieser Änderung an den Trainingsdaten).

Vor Entfernung des Kriteriums „traffic_percent_avg“:

Bezeichnung	**Werte**	
Korrekt klassifizierte Fälle	409	99,5134 %
Falsch klassifizierte Fälle	2	0,4866 %
root mean squared error		0,0697
root relative squared error		14,2946 %

Nach der Entfernung des Kriteriums „traffic_percent_avg“:

Bezeichnung	**Werte**	
Korrekt klassifizierte Fälle	406	98,7835 %
Falsch klassifizierte Fälle	5	1,2165 %
root mean squared error		0,1099
root relative squared error		22,5493 %

Tabelle 4.1: 10-fache stratifizierte Kreuzvalidierung mit und ohne Berücksichtigung des Kriteriums „traffic_percent_avg“

Diese Erhöhung der Fehlerkennzahlen bedeutet gleichzeitig eine Verschlechterung der Erkennungsrate des Klassifikationsalgorithmus J4.8. Diese sinkt von 99,51 auf 98,78 Prozent. Das bedeutet eine relative Verschlechterung um

$$\frac{99,5134 - 98,7835}{100 - 98,7835} = 0,60 = 60\%$$

Das Ergebnis ist also schlechter als vor der Veränderung, obwohl die (inhaltliche) Qualität der Trainingsdaten offensichtlich zugenommen hat. Aus diesem Grund wird im nächsten Abschnitt überprüft, ob nicht ein anderer Klassifikationsalgorithmus besser geeignet ist als der aktuell verwendete J4.8.

4.2 Wahl eines alternativen Algorithmus: RandomTree

Nach der qualitativen Verbesserung der Trainingsdaten, die allerdings paradoxerweise eine quantitative Verschlechterung der Erkennungsleistung ergeben hat, wird in diesem Abschnitt analysiert, inwieweit die Wahl eines alternativen Klassifikationsalgorithmus die Verschlechterung ausgleichen kann. Dazu wurden im Rahmen dieser Arbeit verschiedene anwendbare Algorithmen verglichen, die hier zwar nicht im Einzelnen erläutert, deren Ergebnisse aber ausgewertet werden können. Tabelle 4.2 zeigt ausgewählte Algorithmen und deren Resulta-

te für eine Kreuzvalidierung auf Basis der Trainingsdaten. Eine wesentliche Erkenntnis ist die Tatsache, dass der ursprünglich verwendete Algorithmus offensichtlich (ohne ausführliche Analyse bei Erstellung von Nyx) zu den besseren in dieser Reihe gehört. Deutlich werden vor allem die Unterschiede zu einfachen Verfahren wie „OneR“, bei dem lediglich eine Entscheidungsregel auf Basis der Klasse mit den meisten Elementen erstellt wird. Aber auch komplexere Verfahren wie Bayes'sche Netz scheinen nicht zwangsläufig besser zu sein. Für die aktuell verwendeten Trainingsdaten sind sie sogar schlechter, wie man ebenfalls in der Tabelle sieht. Der einzige Algorithmus, der gleich viele Fälle wie J4.8 korrekt klassifiziert, ist „RandomTree“. Die beiden angegebenen Fehlermaße sind bei diesem Algorithmus sogar noch einmal um einige Prozentpunkte besser.

Algorithmus	**J4.8**	**OneR**	**NaiveBayes**	**KStar**	**BayesNet**	**RandomTree**
korrekt klassifiziert	98,78%	87,10%	90,75%	92,70%	99,51%	**99,51%**
falsch klassifiziert	1,22%	12,90%	9,25%	7,30%	0,49%	**0,49%**
RMSE*	0,11	0,36	0,26	0,22	0,10	**0,07**
RRSE**	22,55%	73,65%	53,06%	44,76%	21,44%	**14,53%**

(* root mean squared error, ** root relative squared error)

Tabelle 4.2: 10-fache stratifizierte Kreuzvalidierung der Trainingsdaten anhand verschiedener Klassifikationsalgorithmen

Der entsprechende Baum für den Algorithmus „RandomTree“, der bei einer Wahl von zwei zufälligen Kriterien zur Konstruktion entsteht, ist im Gegensatz zum J4.8 auch deutlich umfangreicher. Aus diesem Grund sei in Abbildung 4.2 lediglich eine textuelle Darstellung angegeben. Der Aufbau ist hier prinzipiell ähnlich zum Entscheidungsbaum des J4.8-Algorithmus, schließt aber alle verwendeten Kriterien ein. Soll lediglich ein Kriterium zur Konstruktion zufällig gewählt werden, so ändert sich diese Struktur. Das Kriterium „tagged“ steht in der Wurzel, was inhaltlich sogar besser wäre, allerdings ist dann die Erkennungsrate etwas schlechter. Man muss also abwägen, ob eine qualitative oder quantitative Verbesserung gewünscht ist. An dieser Stelle sei letzteres entscheidend, da qualitative Verbesserungen bereits in den vorherigen Abschnitten beschrieben wurden.

Die Frage, ob es einen besseren Klassifikationsalgorithmus gibt, kann also mit diesem direkten Vergleich positiv beantwortet werden. Der Algorithmus „RandomTree“ ist derjenige mit den besten Erkennungsraten und geringsten Fehlermaßen. Der vorher verwendete J4.8 wird deshalb durch „RandomTree“ ersetzt. Dadurch bleibt die qualitative Verbesserung der Trainingsdaten erhalten und gleichzeitig wird die quantitative Verschlechterung ausgeglichen bzw. sogar leicht verbessert.

```
macs_avg < 5.5
|   component_discovered < 0.5
|   |   tagged < 0.5
|   |   |   component_mac_discovered < 0.5
|   |   |   |   custom_criteria1 < 0.5 : false (238/0)
|   |   |   |   custom_criteria1 >= 0.5 : true (1/0)
|   |   |   component_mac_discovered >= 0.5 : true (2/0)
|   |   tagged >= 0.5
|   |   |   macs_avg < 3 : false (3/0)
|   |   |   macs_avg >= 3
|   |   |   |   macs_avg < 4.5
|   |   |   |   |   component_mac_discovered < 0.5 : false (7/0)
|   |   |   |   |   component_mac_discovered >= 0.5 : true (1/0)
|   |   |   |   macs_avg >= 4.5 : false (2/0)
|   component_discovered >= 0.5 : true (48/0)
macs_avg >= 5.5
|   custom_criteria1 < 0.5
|   |   macs_avg < 6.5
|   |   |   component_discovered < 0.5
|   |   |   |   tagged < 0.5 : true (3/0)
|   |   |   |   tagged >= 0.5 : true (13/1)
|   |   |   component_discovered >= 0.5 : true (5/0)
|   |   macs_avg >= 6.5 : true (77/0)
|   custom_criteria1 >= 0.5 : true (11/0)
```

Abbildung 4.2: Textuelle Ausgabe des Entscheidungsbaums durch den Algorithmus „RandomTree“

4.3 Abschlussanalyse der Verbesserungen

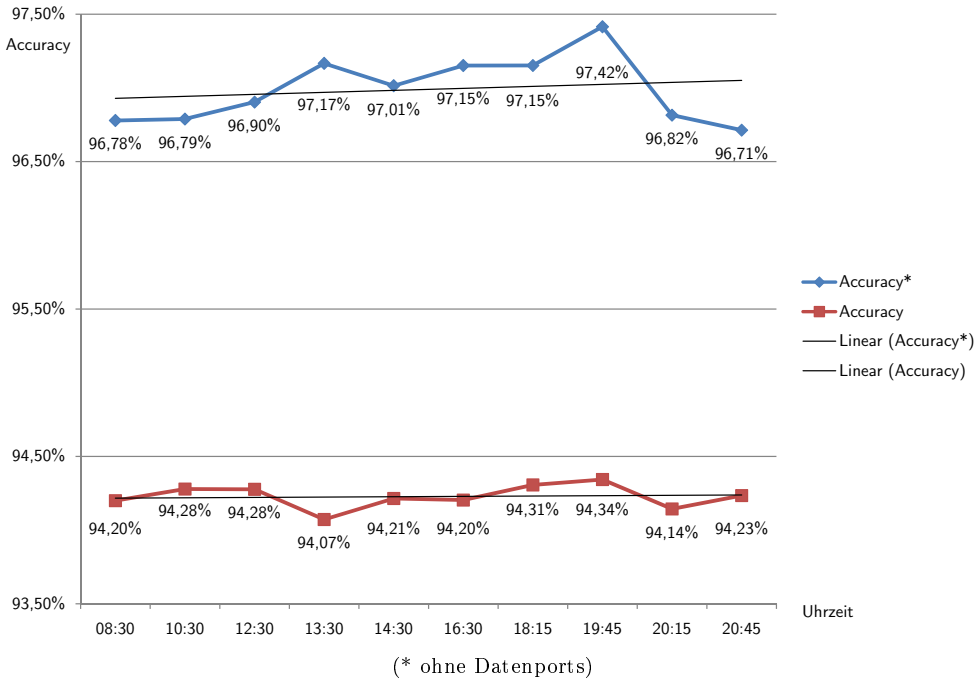
Um die Verbesserungen der vorherigen Abschnitte zusammenfassend beurteilen zu können, kommt wieder das Skript Eos zum Einsatz. Es werden wie zuvor bei der Aufnahme des Ist-Zustands mehrere Durchläufe durchgeführt, die aber zusätzlich zeitlich versetzt sind. So kann sowohl ein gesamteinheitlicher Mittelwert gebildet werden als auch ein tageszeitabhängiger Trend ermittelt werden. Der zweite Punkt ist wichtig, um die Erkennungsleistung in Abhängigkeit von vermuteten tageszeitlichen Veränderungen festzustellen. Damit ergibt sich eine umfassende Analyse, deren Ergebnisse mit dem ermittelten Ist-Zustand vor dieser Arbeit verglichen werden können.

Die Resultate dieser Analyse zeigt Abbildung 4.4 im Detail. Auf den ersten Blick unterscheiden sich diese Werte nicht wesentlich von denen des Ist-Zustands vor dieser Arbeit (siehe Tabelle 3.10, Seite 45). Verdeutlicht man sich aber die Auswirkungen durch die relative Betrachtung, entsprechen sie einem Anstieg um 30 Prozent (im Bereich bis zur absoluten Genauigkeit von 100 Prozent).

$$\text{relative Verbesserung} = \frac{96,99 - 95,72}{100 - 95,72} \approx 0,2967 = 29,67\%$$

Die wesentliche Erkenntnis ist an dieser Stelle aber die Tatsache, dass das System Nyx weiterhin mit einer sehr guten Erkennungsleistung arbeitet. Die Arbeitshypothese von 80 bis 85 Prozent kann deshalb abschließend verworfen werden und durch die neue und gleichzeitig exaktere Hypothese ersetzt werden. Damit liegen zum ersten Mal statistisch abgesicherte Daten zu Leistung von Nyx vor, die weitere Verbesserungen nicht ausschließen.

Als weiteres unbestätigtes Ergebnis dieser Analyse ist festzuhalten, dass sich die Erkennungsleistung möglichweise über den Tag verändert. Das Schaubild 4.3 zeigt nämlich Schwankungen und es gab bereits Vermutungen in diese Richtung. Obwohl die Zahlen nicht repräsentativ sind, scheinen die Werte anzudeuten, dass die Erkennungsrate (ohne Berücksichtigung der Datenports) in gewissem Umfang von der Tageszeit abhängt. Dies bleibt aber als offene Frage dieser Arbeit unbeantwortet und müsste weiter untersucht werden. Die Erkennungsleistung unter Berücksichtigung der Datenports hingegen scheint relativ stabil zu sein. Um hier genauere Aussagen treffen zu können, müsste man ebenfalls deutlich mehr Daten über einen längeren Zeitraum erheben, was im Rahmen dieser Arbeit nicht mehr möglich war.



(* ohne Datenports)

Abbildung 4.3: Diagramm zu den Ergebnisse der Abschlussanalyse durch Eos

Durchlauf	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mittelwert**
Zeitpunkt	**08:30**	**10:30**	**12:30**	**13:30**	**14:30**	**16:30**	**18:15**	**19:45**	**20:15**	**20:45**	
NTP	631	633	626	583	585	614	580	603	608	618	**608**
TP	96,78%	96,79%	96,90%	97,17%	97,01%	97,15%	97,15%	97,42%	96,82%	96,71%	**96,99%**
NFN	21	21	20	17	18	18	17	16	20	21	**19**
FN	3,22%	3,21%	3,10%	2,83%	2,99%	2,85%	2,85%	2,58%	3,18%	3,29%	**3,01%**
NTN	13920	13966	14001	14000	13924	13932	13996	13873	13924	13911	**13945**
TN	94,09%	94,17%	94,16%	93,95%	94,10%	94,08%	94,19%	94,21%	94,03%	94,13%	**94,11%**
NFP	591	580	583	617	589	593	578	569	600	585	**589**
FP	3,99%	3,91%	3,92%	4,14%	3,98%	4,00%	3,89%	3,86%	4,05%	3,96%	**3,97%**
aV	284	285	285	285	284	284	285	283	284	283	**284**
aE	1,92%	1,92%	1,92%	1,91%	1,92%	1,92%	1,92%	1,92%	1,92%	1,91%	**1,92%**
Accuracy*	96,78%	96,79%	96,90%	97,17%	97,01%	97,15%	97,15%	97,42%	96,82%	96,71%	**96,99%**
Accuracy	94,20%	94,28%	94,28%	94,07%	94,21%	94,20%	94,31%	94,34%	94,14%	94,23%	**94,23%**

(* ohne Datenports)

Abbildung 4.4: Ergebnisse der Abschlussanalyse durch Eos

5 Fazit und Ausblick

Fazit

Im Rahmen dieser Bachelorarbeit wurde zunächst dargelegt, wie die Erkennungsrate des Lokalisierungssystems Nyx systematisch und mathematisch fundiert ermittelt werden kann. Dazu wurden verschiedene Verfahren untersucht, wobei sich manuelle Methoden aufgrund der hohen Komplexität des Netzes schnell als ungeeignet herausgestellt haben. Der Schwerpunkt der Analyse lag deshalb im Folgenden auf automatisierbaren Ansätzen. Das Resultat ist – neben einem SQL-basierten Befehl zur Ermittlung der verbundenen Netzkomponenten – das Skript Eos.

Eos ermittelt automatisiert die relevanten Erkennungs- und Fehlerraten, indem zufällig gewählte Ports aus Nyx mit der LRZ Netzdokumentation parametrisierbar verglichen werden. Dabei spiegeln sich die erarbeiteten mathematisch-statistischen Grundlagen in Form von algorithmischen Strukturen exakt wider. Das Ergebnis ist eine unter statistischen Gesichtspunkten präzise Momentaufnahme der Erkennungsleistung von Nyx. Diese wurde vor der Arbeit auf 80 bis 85 Prozent geschätzt, liegt tatsächlich aber etwa 10 Prozentpunkte höher, wie sich bereits bei der Ist-Analyse herausgestellt hat.

Die Ist-Analyse des maschinellen Lernverfahrens hat desweiteren Fehler offenbart, die durch Änderungen am Klassifikationsmodell und an den Trainingsdaten korrigiert wurden. Dadurch wurde sich die Erkennungsleistung qualitativ deutlich verbessert. Die Wahl eines alternativen Lernalgorithmus hat zusätzlich die quantitative Verschlechterung, die (paradoxerweise) zwischenzeitlich durch die Änderungen aufgetreten ist, ausgeglichen. Die Abschlussauswertungen durch Eos bestätigen diese Verbesserungen, da die durchschnittliche Erkennungsrate nun bei etwa 97 Prozent liegt.

Insgesamt wurde also erstmals die Erkennungsleistung des Lokalisierungssystems Nyx ermittelt und Fehler im Klassifikationsmodell durch Korrekturen der Trainingsdaten behoben. Darüber hinaus wird nun ein besserer Klassifikationsalgorithmus verwendet. Zusammenfassend ist damit in Bezug auf das beschriebene Szenario eine bessere Lokalisierung – beispielweise von infizierten oder korrumpierten Rechnern – möglich. Diese können dadurch effektiver identifiziert und Gegenmaßnahmen schneller ergriffen werden.

Ausblick

Mit dieser Bachelorarbeit und der Entwicklung von Eos wurde der Grundstein zur Beurteilung weiterer Verbesserungen gelegt. In Zukunft lassen sich so Veränderungen der Trainingsdaten oder des Lernalgorithmus unmittelbar überprüfen und die Ergebnisse mit den vorherigen Werten vergleichen.

Offen bleibt dabei die Frage, wie die Erkennungsrate der Datenports zu gewichten ist. In dieser Arbeit wurde angenommen, dass Datenports ebenso sicher erkannt werden wie Up- bzw. Downlinkports. Dies konnte empirisch nicht überprüft werden und müsste durch weitere Analysen untermauert werden.

Des Weiteren wurde zwar festgestellt, dass die Erkennungsrate im Tagesverlauf schwankt. Es konnte aber aus zeitlichen Gründen nicht genauer untersucht werden, inwieweit diese Schwankungen charakteristisch für eine gewisse Tageszeit bzw. abhängig vom Netzwerkverkehr zu einer bestimmten Zeit sind.

Außerdem erfolgte die Bewertung von Ports zu Access Points zwar korrekt als Up- bzw. Downlinkport, allerdings erkennt Nyx diese aktuell aus technischen Gründen als Datenports. Sobald Access Points von Nyx korrekt als Netzkomponenten behandelt werden, müsste deshalb eine erneute Analyse mit Eos durchgeführt werden.

Es bleiben also trotz der Ergebnisse dieser Bachelorarbeit weiterhin offene Fragestellungen. Somit stellt diese Arbeit eine Grundlage für weitere Verbesserungen und Untersuchungen des Lokalisierungssystems Nyx dar, die allerdings nun mit Eos beurteilt werden können.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[1] Atteslander, Peter: *Methoden der empirischen Sozialforschung.* Erich Schmidt Verlag, 1991.

[2] Bishop, Christopher M.: *Pattern Recognition And Machine Learning.* Springer, 2006.

[3] Cisco Systems Inc.: *Configuring Cisco Discovery Protocol.* Technischer Bericht, Cisco Systems Inc., 2006. URL: `http://www.cisco.com/en/US/docs/ios/12_2/configfun/configuration/guide/fcf015.pdf`.

[4] Ellenrieder, Peter, Berekoven, Ludwig und Eckert, Werner und: *Marktforschung. Methodische Grundlagen und praktische Anwendung.* Gabler Verlag, 1999.

[5] Group, Network Working: *Introduction and Applicability Statements for Internet Standard Management Framework.* Technischer Bericht, Internet Engineering Task Force (IETF), 2002. URL: `http://tools.ietf.org/html/rfc3410`.

[6] IEEE Std 802.1AB-2005: *Station and Media Access Control Connectivity Discovery.* Technischer Bericht, Institute of Electrical and Electronics Engineers, Inc., 2005. URL: `http://standards.ieee.org/getieee802/download/802.1AB-2005.pdf`.

[7] Kornberger, Ralf und Reiser, Helmut: *"Die Suche nach der Nadel im Heuhaufen-Nyx - Ein System zur Lokalisierung von Rechnern in großen Netzwerken anhand IP- oder MAC-Adressen.* In: *21. DFN Arbeitstagung über Kommunikationsnetze*, Kaiserslautern, Deutschland, Juni 2007. URL: `http://www.nyx.lrz.de/download/Nyx_-_Papier.pdf`.

[8] Kromrey, Dr. Helmut: *Empirische Sozialforschung. Modelle und Methoden der standardisierten Datenerhebung und Datenauswertung.* UTB, Lucius & Lucius Verlag, 1995.

[9] LRZ: *Das Münchner Wissenschaftsnetz – Konzepte, Dienste, Infrastrukturen, Management.* Technischer Bericht, Leibniz-Rechenzentrum, Dezember 2006. URL: `http://www.lrz-muenchen.de/services/netz/mwn-netzkonzept/mwn-netzkonzept.pdf`.

[10] MacKay, David: *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[11] Mayer, Horst Otto: *Interview und schriftliche Befragung: Entwicklung, Durchführung und Auswertung.* Oldenbourg Wissenschaftsverlag, 2008.

[12] Mitchell, Thomas: *Machine Learning.* Mcgraw-Hill International Edit, 1997.

[13] Quinlan, J. Ross: *C4.5: programs for maschine learning.* Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1993.

[14] Quinlan, J. Ross: *Decision trees and multi-valued attributes.* In: J. E. Hayes, D. Michie und J. Richerds (Herausgeber): *Maschine Intelligence 11*, Seiten 305–318. Oxford Press, 1993.

[15] Service, Ferris Research Analyzer Information: *The Global Economic Impact of Spam*, 2005. URL: `http://www.ferris.com/?file_id=2004/05/611_409SpamCosts.pdf`.

[16] Shannon, C. E.: *A Mathematical Theory of Communication.* B&T, Dezember 1963.

[17] Steinmüller, Dr. Johannes: *Maschinelles Lernen – Vorlesung an der TU Chemnitz, Wintersemester 2006/2007*, 2006. URL: `http://www-user.tu-chemnitz.de/~stj/lehre/masch.pdf`.

[18] Stone, P. J., Hunt, E. B. und Marin, J. und: *Experiments in Induction.* Academic Press, New York, 1966.

[19] Taylor, C. C., Michie, D. und Spiegelhalter, D. J. und: *Machine Learning, neural and statistical Classification.* Ellis Horwood Series in Artificial Intelligence, 1994.

[20] Witten, Ian H. und Frank, Eibe: *Data mining: practical machine learning tools and techniques.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, USA, 2. Auflage, 2005. ISBN 0-12-088407-0. URL: `http://www.cs.waikato.ac.nz/~ml/weka/`.

A Hilfsmittel

A.1 Eingesetzte Software

PuTTY ist ein freier SSH- und Telnet-Client. SSH steht für Secure Shell und ermöglicht es (wie Telnet) sich an einem anderen Rechner interaktiv einzuloggen (vorausgesetzt natürlich, dass man an diesem Rechner eine entsprechende Kennung hat). Wesentliche Vorteile gegenüber Telnet sind, dass alle Daten verschlüsselt übertragen werden und dass eine strenge Authentifizierung beider beteiligten Parteien stattfindet (in Anlehnung an `http://www.lrz-muenchen.de/services/security/putty/`).

HeidiSQL (vorher bekannt als MySQL-Front) ist ein freier Open Source Client für MySQL. HeidiSQL diente als Front-End für die diversen Nyx-Datenbanken auf unterschiedlichen Servern („deb-nyxtestdb“ [10.156.10.84] und „lxsql99.lrz-muenchen.de“ [10.156.10.180]).

TOAD (for Oracle) steht für „Tool For Application Developers“ und ist ein grafisches Front-End für Oracle-Datenbanken mit Schwerpunkt auf den drei Anwendungsbereichen PL/SQL Programmentwicklung, Datenbank-Administration und Datenanalyse. Die verwendete Konfiguration ist im übernächsten Abschnitt gesondert aufgeführt.

Eclipse ist ein Open-Source-Framework und eine unter Programmierern bekannte Entwicklungsumgebung (IDE) zur Erstellung von Software nahezu aller Art. Erweitert wird Eclipse unter anderem durch das Projekt EPIC (Eclipse Perl Integration), das einen komfortablen Perl-Editor bietet. Dieser wurde im Rahmen der Bachelorarbeit in Kombination mit der Perl-Distribution „ActivePerl“ zur Erstellung des Analyseskripts Eos verwendet.

Waikato Environment for Knowledge Analysis (Weka) (siehe Abschnitt 2.2.2)

LaTeX ist ein Softwarepaket, das die Benutzung des Textsatzprogramms TEXmit Hilfe von Makros vereinfacht. LATEX (pdflatex) wurde verwendet, um diese Bachelorarbeit zu setzen.

A.2 Konfiguration von TOAD mittels Oracle TNS Name

```
ORACLE9I =
(DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = localhost)
        (PORT = 1521)
      )
    )
    (CONNECT_DATA =
      (SERVICE_NAME = oracle9i)
    )
  )
```

B Trainingsdaten

Ausschnitt aus „seed.arff“ vor dieser Arbeit:

```
@RELATION downlinkport
@ATTRIBUTE macs_avg numeric
@ATTRIBUTE macs_avg_cnt numeric
@ATTRIBUTE traffic_percent_avg numeric
@ATTRIBUTE component_mac_discovered numeric
@ATTRIBUTE component_discovered numeric
@ATTRIBUTE tagged numeric
@ATTRIBUTE custom_criteria1 numeric
@ATTRIBUTE class true,false
@DATA
474, 7, 0, 1, 0, 1, 0, true
20, 16431, 45, 0, 1, 0, 0, true
188, 4, 0, 1, 1, 1, 0, true
2, 17851, 14, 0, 0, 0, 0, false
4, 17850, 11, 0, 0, 0, 0, false
31, 17569, 0, 0, 1, 0, 0, true
0, 4, 2, 0, 0, 0, 0, false
1, 17847, 1, 0, 1, 0, 0, true
2, 17842, 4, 0, 0, 0, 0, false
8, 4, 0, 0, 1, 1, 1, true
6, 4, 17, 0, 1, 1, 0, true
1, 7, 2, 0, 0, 0, 0, false
1, 5, 2, 0, 0, 0, 0, false
0, 4, 0, 1, 0, 0, 0, true
593, 7, 0, 1, 0, 1, 0, true
...
```

Ausschnitt aus „seed.arff“ nach dieser Arbeit (hier wurden die beiden Kriterien „macs_avg_cnt“ und „traffic_percent_avg“ entfernt):

```
@RELATION downlinkport
@ATTRIBUTE macs_avg numeric
@ATTRIBUTE component_mac_discovered numeric
@ATTRIBUTE component_discovered numeric
@ATTRIBUTE tagged numeric
@ATTRIBUTE custom_criteria1 numeric
@ATTRIBUTE class true,false
@DATA
474, 1, 0, 1, 0, true
20, 0, 1, 0, 0, true
188, 1, 1, 1, 0, true
2, 0, 0, 0, 0, false
4, 0, 0, 0, 0, false
31, 0, 1, 0, 0, true
0, 0, 0, 0, 0, false
1, 0, 1, 0, 0, true
2, 0, 0, 0, 0, false
8, 0, 1, 1, 1, true
6, 0, 1, 1, 0, true
1, 0, 0, 0, 0, false
1, 0, 0, 0, 0, false
0, 1, 0, 0, 0, true
593, 1, 0, 1, 0, true
...
```

C Eos: Quellcode

eos.pl

```
1   #!/usr/bin/perl
2   use strict;
3   use Getopt::Long;
4   use DBI;
5
6   # Deactivate buffer for STDOUT
7   select STDOUT;
8   $| = 1;
9
10
11  # VERSION ###############################################################
12  # Version Number: 1.0.6
13  ########################################################################################
14
15
16  # CONFIG ################################################################
17
18  # TestNyx
19  #my $nyx_host = "10.156.10.84";
20  #my $nyx_port = 3306;
21  #my $nyx_database_name = "nyx2";
22  #my $nyx_username = "nyx";
23  #my $nyx_password = "nyxnyx";
24
25  # ProduktionsNyx
26  my $nyx_host = "10.156.10.180"; # = lxsql199.lrz-muenchen.de
27  my $nyx_port = "3306";
28  my $nyx_database_name = "nyx-1-6";
29  my $nyx_username = "nyx";
30  my $nyx_password = "nyxnyx";
31
32
33  # Netzdoku
34  my $nd_host = "10.156.8.13"; # tip: host = w2ksrv14.lrz-muenchen.de = 10.156.8.13
35  my $nd_port = 1521;
36  my $nd_sid = "oracle9i";
37  my $nd_username = "ndreadonly";
38  my $nd_password = "ndRO&2";
39
40  # END: CONFIG ###########################################################
41
42
43  # basic variables
44  my $samples = -1;
45  my $progress = 1;
46  my $backspaces = "";
47  my $consider_auto_updownlink_override = 0; #false
48  my $accesspoints_as_updownlink = 0;
49  my $dataport_accuracy_loading = -1;
50  my $key = 0; # false
51  my $key_only = 0; # false
52
53  # Prototype (-> subroutines at the end of this script)
54  sub help;
55
56  # arguments
57  GetOptions( 's=i' => \$samples,
58              's=i' => \$samples,
59
```

```
60              'dataport_loading=f' => \$dataport_accuracy_loading,
61              'dl=f' => \$dataport_accuracy_loading,
62
63              'heuristics' => \$consider_auto_updownlink_override,
64              'h' => \$consider_auto_updownlink_override,
65
66              'ap' => \$accesspoints_as_updownlink,
67
68              'key' => \$key,
69
70              'keyonly' => \$key_only,
71
72              'help|?' => \&help) ;
73
74  if ($key_only eq 1) {
75      my $result = &print_key();
76      print $result."\n";
77      exit 0;
78  }
79
80  help unless ($samples > 0 and $dataport_accuracy_loading >= 0 and $dataport_accuracy_loading<=1);
81
82
83
84  # DATABASES - CONNECT ############################################################
85  my $dbh_mysql;
86  my $dbh_oracle;
87
88  $dbh_mysql =
89  DBI->connect('DBI:mysql:database='.$nyx_database_name.';host='.$nyx_host.';port='.$nyx_port,
    $nyx_username, $nyx_password) ||
90      die "Couldn't connect to database: " . DBI->errstr;
91
92  $dbh_oracle =
93  DBI->connect('DBI:Oracle:host='.$nd_host.';sid='.$nd_sid.';port='.$nd_port, $nd_username,
    $nd_password) ||
94      die "Couldn't connect to database: " . DBI->errstr;
95
96  # END: DATABASES - CONNECT ########################################################
97
98
99
100 # NYX ##############################################################################
101
102 my $sql="
103     SELECT  interfaceid, updownlink, ifdescription,
104             man_updownlink_override,auto_updownlink_override
105     FROM    interface, device
106     WHERE   interface.device = device.id
107         AND device.faultcount < 100
108     ORDER BY RAND()
109     LIMIT $samples
110 ";
111
112 my $sth_mysql = $dbh_mysql->prepare ($sql);
113 $sth_mysql->execute() || die "Couldn't execute statement: " . $sth_mysql->errstr;
114
115 # (results from NYX)
116 my @interfaceids;       # list of ids (array)
117 my %ifnames;            # interfaceid->ifdescr (hash)
118 my %devices;            # interfaceid->sysname (hash)
119 my %reversed_devices;   # sysname->interfaceid (hash)
120 my %classify;           # interfaceid->updownlink (hash)
121
122 my $u = 0;              # manual or auto override to updownlink
123 my $d = 0;              # manual or auto override to dataport
124 my $num_ifs = 0;        # current number of interface
125 my $num_sysnames = 0;   # number of sysnames got from NYX
126
127 print "NYX:\n";
128
129 #####################################
130 # Step 1:
131 #########
132 # get interfaceids and classify status
133 while (my @data = $sth_mysql->fetchrow_array()) {
134     chomp $data[0];     # interfaceid
135     chomp $data[1];     # updownlink
136     chomp $data[2];     # ifdesc
137     chomp $data[3];     # man_updownlink_override
```

```
138       chomp $data[4];     # auto_updownlink_override
139
140       # array of interfaceids gets the interfaceid from NYX in lower cases
141       $interfaceids[$num_ifs] = lc $data[0];
142
143       # hash of interface names gets the ifnames from NYX in lower cases
144       $ifnames{lc $data[0]} = lc $data[2];
145
146       # if we have manual or auto override to updownlink
147       if ( (($data[3] eq "1") or ($data[4] eq "1")) and $consider_auto_updownlink_override eq 1)
148       {
149           $classify{lc $data[0]}=1;
150           $u++;
151       }
152       # if we have manual or auto override to dataport
153       else {
154           if  ( (($data[3] eq "0") or ($data[4] eq "0"))and $consider_auto_updownlink_override eq 1)  {
155               $classify{lc $data[0]}=0;
156               $d++;
157           }
158           else {
159               # if we have NO manual or auto override
160               if ( ($data[3] eq "-1") and ($data[4] eq "-1")) {
161                   $classify{lc $data[0]}=$data[1]; # get what Nyx said
162           }
163       }
164   }
165    $num_ifs++;
166   }
167   print "\tInterfaces: $num_ifs\n";
168   if ($consider_auto_updownlink_override eq 1) {
169       print "\tUpdownlinks because of manual or auto override: $u\n";
170       print "\tDataports   because of manual or auto override: $d\n";
171   } else {
172       print "\tHeuristics (briding firewall) and manual classification not considered\n";
173       print "\t (= manual and auto updownlink override are extraneous)\n"
174   }
175
176   ######################################
177   # Step 2:
178   #########
179   # switchnames (sysname) for interfaceids
180   my $i=0;
181
182   $sql = "SELECT  sysname, interfaceid
183           FROM    device, interface
184           WHERE   device.id=interface.device";
185
186   $sth_mysql = $dbh_mysql->prepare ($sql);
187   $sth_mysql->execute() || die "Couldn't execute statement: " . $sth_mysql->errstr;
188
189   while (my @data = $sth_mysql->fetchrow_array()) {
190       chomp $data[0];     # sysname
191       chomp $data[1];     # interfaceid
192
193       $devices{lc $data[1]}=lc $data[0];              #interfaceid->sysname
194       $reversed_devices{lc $data[0]}=lc $data[1];     #sysname->interfaceid
195       $num_sysnames++;
196   }
197
198   print "\tNumber of sysnames: $num_sysnames\n";
199   print "\n";
200
201   # END: NYX ###############################################################################
202
203
204
205   # NETZDOKU ###############################################################################
206   my %nd_devices;                     # Komponente: id->alias (hash)
207   my %nd_bezeichnungen;               # PhyPort: (if)id->Bezeichnung (hash)
208   my %nd_interfaces;                  # phyport.id -> phyport.komponente_id (hash)
209   my %nd_connected_via_line;          # phyport.id -> phyport.leitung_id (hash)
210   my %nd_connected_to_if;             # phyport.id -> connected_phyport_id (hash)
211   my %nd_device_alias_from_portid;    # phyport.id -> Komponente.alias
212   my @nd_ids;                         # list of netzdoku ids (array)
213
214   my $num_nd_updwns = 0;
215
216   ######################################
217   # Step 1:
```

```
218  #########
219  # get Komponente.alias for Komponente.id
220  $sql = "SELECT  alias, id
221          FROM    Netzdoku.Komponente";
222  my $sth_oracle = $dbh_oracle->prepare($sql) ||
223      die "Couldn't prepare statement: " . $dbh_oracle->errstr;
224
225  $sth_oracle->execute() ||
226      die "Couldn't execute statement: " . $sth_oracle->errstr;
227
228  $i=0;
229  while (my @data = $sth_oracle->fetchrow_array()) {
230      chomp $data[0]; # alias
231      chomp $data[1]; # id
232
233      # !!! Komponent: id -> alias !!!
234      $nd_devices{$data[1]}=lc $data[0];
235
236      $i++;
237
238  }
239  print "NETZDOKU:\n";
240  print "\tNumber of components: $i\n";
241
242  ######################################
243  # Step 2:
244  #########
245  # get perl hashes of the NETZDOKU.PhyPort table
246  $sql = "SELECT  bezeichnung, komponente_id, id, leitung_id
247          FROM    Netzdoku.PhyPort";
248  $sth_oracle = $dbh_oracle->prepare($sql) ||
249      die "Couldn't prepare statement: " . $dbh_oracle->errstr;
250  $sth_oracle->execute() ||
251      die "Couldn't execute statement: " . $sth_oracle->errstr;
252
253  while (my @data = $sth_oracle->fetchrow_array()) {
254      chomp $data[0]; # bezeichnung
255      chomp $data[1]; # kompnente_id
256      chomp $data[2]; # id
257      chomp $data[3]; # leitung_id
258
259      # !!! phyport.id -> phyport.bezeichnung !!!
260      $nd_bezeichnungen{$data[2]}=lc $data[0];
261
262      # !!! phyport.id -> phyport.komponente_id !!!
263      $nd_interfaces{$data[2]}=$data[1];
264
265      # !!! phyport.id in array !!!
266      $nd_ids[$num_nd_updwns]=$data[2];
267
268      # !!! phyport.id -> phyport.leitung_id
269      $nd_connected_via_line{$data[2]}=$data[3];
270
271      $num_nd_updwns++;
272  }
273  print "\tNumber of PhyPorts: $num_nd_updwns\n";
274
275  ######################################
276  # Step 3:
277  ##########
278  # get the phyport.id for the connected port by the leitung.id
279  while (my ($nd_phyport_id, $nd_conn_phyport_line_id)=each(%nd_connected_via_line)) {
280
281      # line id 0 means "not connected" to another component,
282      # so we can ignore all offline ports.
283      # line id geater than 0 represents an existing connection
284      # and that is why a particualar line id exists exactly twice
285      # (and we search for the other port id which is not the own id)
286      $sql = "SELECT  id
287              FROM    Netzdoku.PhyPort
288              WHERE   leitung_id<>'0'
289                      and leitung_id='".$nd_conn_phyport_line_id."'
290                      and id<>'".$nd_phyport_id."'";
291
292      $sth_oracle = $dbh_oracle->prepare($sql) ||
293          die "Couldn't prepare statement: " . $dbh_oracle->errstr;
294
295      $sth_oracle->execute() || die "Couldn't execute statement: " . $sth_oracle->errstr;
296
297      while (my @data = $sth_oracle->fetchrow_array()) {
```

```
298             chomp $data[0];
299             # !!! phyport.id -> (connected) phyport.id !!!
300             $nd_connected_to_if{$nd_phyport_id}=$data[0];
301         }
302     }
303
304     ######################################
305     # Step 4:
306     ##########
307     # get Komponete.alias for a port.id: port.id -> Komponente.alias
308     $sql = "SELECT  phyport.id, komponente.alias
309             FROM    Netzdoku.komponente, Netzdoku.phyport
310             WHERE   komponente.id = phyport.komponente_id";
311         $sth_oracle = $dbh_oracle->prepare($sql) ||
312             die "Couldn't prepare statement: " . $dbh_oracle->errstr;
313         $sth_oracle->execute() ||
314             die "Couldn't execute statement: " . $sth_oracle->errstr;
315
316     while (my @data = $sth_oracle->fetchrow_array()) {
317         chomp $data[0]; # id
318         chomp $data[1]; # alias
319         $nd_device_alias_from_portid{$data[0]}=lc $data[1];
320     }
321
322     #
323     print "\n";
324     # END: NETZDOKU ######################################################################
325
326
327
328     # COMPARISON ###########################################################################
329     my $nyx_updownlinkcount=0;
330     my $nyx_false_negative=0;
331     my $nyx_false_positive=0;
332     my $nyx_dataport=0;
333     my $not_found_in_nd=0;
334     my $nd_count=0;
335     my $heuristically_recognized_ports=0;
336     my $nyx_if_found_in_nd="false";
337     my $nyx_nd_sysname_match="false";
338
339
340     ######################################
341     # Directory and file handling:
342     ##########
343     my $mkdir_ok = 0; # false
344     my $mkdir_sn = 0; # additional number, if dir already exists
345     my %time=&get_time;
346     my $mkdir_directory_pattern = "results_"
347         .(sprintf "%04d",$time{year}).(sprintf "%02d",$time{month}).(sprintf "%02d",$time{day})
348         ."_".(sprintf "%02d",$time{hour}).".-".(sprintf "%02d",$time{minute});
349     my $mkdir_directory = $mkdir_directory_pattern;
350
351     # reliable creation of a result directory
352     while ($mkdir_ok eq 0) {
353         $mkdir_ok = mkdir $mkdir_directory, 0777;
354         if ($mkdir_ok eq 0) {
355             $mkdir_directory = $mkdir_directory_pattern . "_" .$mkdir_sn;
356             $mkdir_sn++; # increase number
357         }
358     }
359
360     print "Directory '$mkdir_directory' created for results.\n\n";
361
362     chdir ("./".$mkdir_directory);
363
364     open RECOG,     ">recognized_ports.txt";
365     open NEGATIVES, ">false_negatives.txt";
366     open POSITIVES, ">false_positives.txt";
367     open DATAPORTS, ">dataports.txt";
368     open NOSWITCH,  ">switch_not_found.txt";
369     open HEURISTIC, ">heuristically_recognized.txt";
370     open RESULT,    ">eos_results.txt";
371
372     my $output_headline =
373     "NYX_SYSNAME; NYX_INTERFACEID; NYX_IFDESCRIPTION; NETZDOKU_ALIAS; NETZDOKU_BEZEICHNUNG;
        CLASSIFICATION; COMMENT\n";
374
375     print RECOG     $output_headline;
376     print NEGATIVES $output_headline;
```

```
377  print POSITIVES $output_headline;
378  print DATAPORTS $output_headline;
379  print NOSWITCH  $output_headline;
380  print HEURISTIC $output_headline;
381
382
383  ####################################
384  # Comparison:
385  ##########
386  print "Comparison of interfaces started: "; # + progression
387  $progress = "";
388  $backspaces = "";
389
390  my $done = 0;
391
392  #go through all Nyx interfaces
393  for my $nyx_cur_if (@interfaceids) {
394
395      # progress
396      ($progress, $backspaces) = progress ($backspaces, $progress);
397
398      # get properties of the interface from NYX.interface
399      my $nyx_cur_ifdescr=$ifnames{$nyx_cur_if};      # ifdescription
400      my $nyx_cur_sysname=$devices{$nyx_cur_if};      # sysname
401      my $nyx_cur_if_classify=$classify{$nyx_cur_if}; # classification
402
403      # check if sysname it NOT empty and the length (ifdescr)<=3 -> e.g. "A12"
404      if ( (!($nyx_cur_sysname eq "")) and (length $nyx_cur_ifdescr <=3)) {
405
406          # we ignore routers because they are ALWAYS (per default) classified as up-/downlinks
407          if (!(substr ($nyx_cur_sysname,0,3) eq "csr")) {
408              $nyx_nd_sysname_match="false";
409
410              # STEP 1:
411              # Assumption:
412              # If an interface an its component is in the NETZDOKU and the properties
413              # discovered by NYX match the properties recorded in the NETZDOKU,
414              # the information is approved.
415              # Consequence:
416              # Interfaces with classification eq "1" are recognized correctly,
417              # others are not recognized correcty by NYX except the connected
418              # device/component is no a switch
419
420              # go through all Netzdoku aliases
421              while (my ($nd_cur_komponente_id, my $nd_cur_komponente_alias) = each (%nd_devices)) {
422                  # %nd_devices: Komponente:id->alias
423
424                  # if the NXY.sysnames match ND.aliases
425                  if ($nyx_cur_sysname eq  $nd_cur_komponente_alias) {
426
427                      $nyx_nd_sysname_match="true";
428                      $nd_count=0;
429
430                      # go through all NETZDOKU interfaces
431                      while (my ($nd_cur_phyportid, my $nd_cur_komponente_id_tmp) = each (
%nd_interfaces)) {
432                          #my %nd_interfaces; # phyport.id -> komponente_id
433
434                          $nd_count++;
435
436                          # if the device ids match
437                          if ($nd_cur_komponente_id_tmp eq $nd_cur_komponente_id){
438                              # get current NETZDOKU.phyport.bezeichnung
439                              my $nd_cur_bez=$nd_bezeichnungen{$nd_cur_phyportid};
440
441                              # if NETZDOKU.phyport.bezeichnung ^= NYX.ifdescription, e.g. "A12"
442                              if ($nd_cur_bez eq $nyx_cur_ifdescr) {
443
444                                  $nyx_if_found_in_nd="true"; # set the found marker
445
446                                  # only if interface is not connected to a known switch, etc. ->
substring
447                                  my $alias_substring = substr($nd_device_alias_from_portid{
$nd_connected_to_if{$nd_cur_phyportid}}, 0, 2);
448                                  my $ap_boolean = 0;
449                                  if ($alias_substring eq "ap" and $accesspoints_as_updownlink eq 1) {
450                                      $ap_boolean = 1;
451                                  }
452
453                                  # if the port has been classfied correctly as uplownlink
```

```
454                                 if ($nyx_cur_if_classify eq "1") {
455                                     $nyx_updownlinkcount++;
456                                     print RECOG "$nyx_cur_sysname;
    $reversed_devices{$nyx_cur_sysname}; $nyx_cur_ifdescr; $nd_cur_komponente_alias; $nd_cur_bez;
    $nyx_cur_if_classify; RECOGNIZED\n";
457                                     $done = 1;
458                                 }
459                                 elsif ($nyx_cur_if_classify eq "0" and
460                                       ($alias_substring eq "sw" ||
461                                        $alias_substring eq "cs" ||
462                                        $alias_substring eq "f5" ||
463                                        $alias_substring eq "ip" ||
464                                        $alias_substring eq "wd" ||
465                                             $ap_boolean eq 1)) {
466
467                                     $nyx_false_negative++;
468                                     print NEGATIVES "$nyx_cur_sysname;
    $reversed_devices{$nyx_cur_sysname}; $nyx_cur_ifdescr; $nd_cur_komponente_alias; $nd_cur_bez;
    $nyx_cur_if_classify; FALSE_NEGATIVE\n";
469                                     $done = 1;
470                                 }
471
472                             }#end match nyx if descr
473                         }#end match dev ids
474
475                         # if interface already successful compared
476                         if ($done eq 1) {
477                             last; # = break in perl
478                         }
479
480                     }#end while ifs
481                 }#end of match sysname
482
483             # if device and interface already successful compared
484             if ($done eq 1) {
485                 last; # = break in perl
486             }
487
488             }#end of while nd_devices
489
490
491
492             # STEP 2:
493             # Assumption:
494             # If the switch is a valid one (in NETZDOKU), but the interface is not
495             # found in the NETZDOKU, this interface is a dataport.
496             # Consequence:
497             # Interfaces with classification as dataport (=0) are recognized correctly,
498             # others are false positives except for the heuristically recognized ones
499
500
501             # we had a match (NYX.sysname == NETZDOKU.Komponente.alias) before,
502             # so we have a valid switch
503             if ($nyx_nd_sysname_match eq "true" and $done ne 1) {
504
505                 # the interface is in NYX but NOT in NETZDOKU
506                 if ($nyx_if_found_in_nd eq "false") {
507                     #check classification
508
509                     # if the port has been classfied as uplownlink
510                     # => false postive (or perhaps a briding firewall port)
511
512                     if ($nyx_cur_if_classify eq "1") {
513
514                         # check if it is a briding firewall port
515                         my $sql="SELECT auto_updownlink_override
516                                  FROM interface
517                                  WHERE interfaceid='$nyx_cur_if'";
518                         my $sth_mysql = $dbh_mysql->prepare ($sql);
519                         $sth_mysql->execute() ||
520                             die "Couldn't execute statement: " . $sth_mysql->errstr;
521
522                         my @auto_updownlink_override = $sth_mysql->fetchrow_array();
523
524                         chomp $auto_updownlink_override[0];
525
526                             # autoveride==1 => bridging firewall port
527                             if ($auto_updownlink_override[0] eq "1") {
528                                 # $nyx_cur_sysname equals NETZDOKU.Komponente.alias,
529                                 # so we write it to the export file
```

```
530                                 print HEURISTIC  "$nyx_cur_sysname;
    $reversed_devices{$nyx_cur_sysname}; $nyx_cur_ifdescr;$nyx_cur_sysname;;$nyx_cur_if_classify;
    BR_FW_PORT\n";
531                                 $heuristically_recognized_ports++;
532                                 $done = 1;
533                             }
534                             else { # false positive
535                                 # $nyx_cur_sysname equals NETZDOKU.Komponente.alias,
536                                 # so we write it to the export file
537                                 print POSITIVES "$nyx_cur_sysname;
    $reversed_devices{$nyx_cur_sysname}; $nyx_cur_ifdescr;$nyx_cur_sysname;;$nyx_cur_if_classify;
    FALSE_POSITIVE\n";
538                                 $nyx_false_positive++;
539                                 $done = 1;
540                             }
541                     }
542                     else {
543                         print DATAPORTS
    "$nyx_cur_sysname;$reversed_devices{$nyx_cur_sysname};$nyx_cur_ifdescr;;;$nyx_cur_if_classify;
    DATAPORT\n";
544                         $nyx_dataport++;
545                         $done = 1;
546                     }
547                 }
548             }
549             elsif ($done ne 1) { # switches we had NO MATCH: $nyx_nd_sysname_match eq "false"
550                 print NOSWITCH "$nyx_cur_sysname;$nyx_cur_ifdescr;;;$nyx_cur_if_classify;
    SWITCH_NOT_FOUND\n";
551                 $not_found_in_nd++;
552             }
553
554             # reset
555             $done = 0;
556             $nyx_if_found_in_nd="false";
557
558         } # end if ignore routers
559     } # end if sysname !=""
560 } # end of for
561
562 progress ($backspaces, $progress);
563 print "\n\n\n";
564 # END: COMPARISON ################################################################
565
566
567
568 # FILE HANDLER - CLOSE ###########################################################
569
570 close RECOG;
571 close NEGATIVES;
572 close POSITIVES;
573 close DATAPORTS;
574 close NOSWITCH;
575 close HEURISTIC;
576 # close RESULT comes later in script #
577
578 # END: FILE HANDLER - CLOSE ######################################################
579
580
581
582 # CALCULATIONS ###################################################################
583 my $adjustmentValue = int($nyx_dataport * (1 -$dataport_accuracy_loading));
584 $nyx_dataport = $nyx_dataport - $adjustmentValue;
585
586 my $samples_considered = int( $nyx_updownlinkcount
587                             + $nyx_false_negative
588                             + $nyx_false_positive
589                             + $nyx_dataport
590                             + $adjustmentValue);
591
592 my $samples_valid = $samples_considered + $heuristically_recognized_ports + $not_found_in_nd;
593 my $samples_invalid = $samples - $samples_valid;
594 my $false_negative_percent = 0;
595 my $false_positive_percent = 0;
596 my $true_positive_percent = 0;
597 my $true_negative_percent = 0;
598 my $adjustmentError = 0;
599
600 # false negative and false positive
601 if ($nyx_false_negative + $nyx_updownlinkcount ne 0) {
602     $false_negative_percent = $nyx_false_negative/($nyx_false_negative + $nyx_updownlinkcount);
```

```
603 }
604
605 if (($nyx_false_positive + $nyx_dataport) ne 0) {
606     $false_positive_percent = $nyx_false_positive/($nyx_false_positive + $nyx_dataport +
    $adjustmentValue);
607 }
608
609 # true negative (sensibility) and true positive (recall or sensitivity)
610 if (($nyx_updownlinkcount + $nyx_false_negative) ne 0) {
611     $true_positive_percent = $nyx_updownlinkcount/($nyx_updownlinkcount + $nyx_false_negative);
612 } elsif ($nyx_updownlinkcount eq 0 and $nyx_false_negative eq 0) {
613     $true_positive_percent = 1;
614 }
615
616 if (($nyx_false_positive + $nyx_dataport + $adjustmentValue) ne 0) {
617     $true_negative_percent = $nyx_dataport/($nyx_false_positive + $nyx_dataport + $adjustmentValue);
618 } elsif ($nyx_false_positive eq 0 and $nyx_dataport eq 0 and $adjustmentValue eq 0) {
619     $true_negative_percent = 1;
620 }
621
622 # adjustment
623 if (($nyx_false_positive + $nyx_dataport + $adjustmentValue) ne 0) {
624     $adjustmentError = $adjustmentValue/($nyx_false_positive + $nyx_dataport + $adjustmentValue);
625 } elsif ($nyx_false_positive eq 0 and $nyx_dataport eq 0 and $adjustmentValue eq 0) {
626     $adjustmentError = 0;
627 }
628
629 # accuracy
630 my $accuracy_with_dataports = ($nyx_updownlinkcount + $nyx_dataport)/ $samples_considered;
631 my $accuracy_without_dataports = $true_positive_percent;
632
633 # minimal number of samples to get an acceptable result
634 my $c_N = $num_sysnames;
635 my $c_alpha1 = 0.05;  # alpha1
636 my $c_alpha2 = 0.01;  # alpha2
637 my $c_t1 = 1.96;      # z-quantile: z_(1-(c_alpha1/2)) = z_(0.9725)
638 my $c_t2 = 2.58;      # z-quantile: z_(1-(c_alpha2/2)) = z_(0.9950)
639 my $c_p = 0.5;        # maximum
640 my $c_n1 = ($c_t1*$c_t1  * $c_N * $c_p * (1-$c_p)) / ($c_t1*$c_t1 * $c_p * (1-$c_p) + $c_alpha1*
    $c_alpha1 * ($c_N - 1));
641 my $c_n2 = ($c_t2*$c_t2  * $c_N * $c_p * (1-$c_p)) / ($c_t2*$c_t2 * $c_p * (1-$c_p) + $c_alpha2*
    $c_alpha2 * ($c_N - 1));
642 my $correct1 = 1-$c_alpha1;
643 my $correct2 = 1-$c_alpha2;
644
645 # END: CALCULATIONS ############################################################
646
647
648 # RESULT STRING FOR EOS ########################################################
649 my $line = "--------------------------------------------------------------------------------\n";
650
651 my $result_string = "";
652 $result_string .= "EOS results:\n";
653 $result_string .= $line;
654
655 $result_string .= "Maximum number of samples tested                      : $samples\n";
656 $result_string .= "|\n";
657 $result_string .= "|- Invalid switches (or residual value to maximum)      : $samples_invalid\n";
658 $result_string .= "|\n";
659 $result_string .= "|- Valid switches (sysname not empty, ifdescription < 3): $samples_valid\n";
660 $result_string .= "   |\n";
661 $result_string .= "   |- Switches not found in NETZDOKU (not considered)   : $not_found_in_nd\n";
662 $result_string .= "   |\n";
663 $result_string .= "   |- Heuristically recognized ports (not considered)   :
    $heuristically_recognized_ports\n";
664 $result_string .= "   |\n";
665 $result_string .= "   |- Number of considered samples                      : $samples_considered\n";
666 $result_string .= "      |\n";
667 $result_string .= "      |- Number of Updownlinkports                      : $nyx_updownlinkcount\n";
668 $result_string .=
669         sprintf  ("         |- True positive ratio: TP = %.02f\%\n", $true_positive_percent*100);
670 $result_string .= "      |\n";
671 $result_string .= "      |- Number of FN: Updownlink classified as dataport: $nyx_false_negative\n";
672 $result_string .=
673         sprintf  ("         |- False negatives ratio: FN = %.02f\%\n", $false_negative_percent*100);
674 $result_string .= "      |\n";
675 if ($dataport_accuracy_loading eq 1) {
676 $result_string .= "      |- Number of Dataports                            : $nyx_dataport\n";
677 } else {
678 $result_string .= "      |- Adjusted Number of Dataports                   : $nyx_dataport\n";
```

```
679  }
680  $result_string .=
681          sprintf  ("         |- True negatives ratio: TN = %.02f\%\n", $true_negative_percent*100);
682  $result_string .= "      |\n";
683  $result_string .= "      |- Number of FP: Dataport classified as updownlink: $nyx_false_positive\n";
684  $result_string .=
685          sprintf  ("         |- False positive ratio: FP = %.02f\%\n", $false_positive_percent*100);
686  $result_string .= "      |\n";
687  if ($dataport_accuracy_loading ne 1) {
688  $result_string .= "      |- Adjustment value                                : $adjustmentValue\n";
689  $result_string .=
690          sprintf  ("         |- Adjustment Error: AE = %.02f\%\n", $adjustmentError*100);
691  $result_string .= "      |\n";
692  }
693  $result_string .=
694          sprintf  ("      |- Accuracy  (no dataports included)              : %.02f\%\n",
     $accuracy_without_dataports*100);
695  $result_string .=
696          sprintf  ("      |- Accuracy  (dataports included)                 : %.02f\%\n",
     $accuracy_with_dataports*100);
697
698  $result_string .= $line;
699  $result_string .= "\n";
700
701  # output of statistical calculations
702  if ($samples_considered > $c_n2) {
703      $result_string .=
704          sprintf ("(Results are correct with %.02f percent confidence)\n", $correct2*100);
705  } elsif ($samples_considered > $c_n1) {
706      $result_string .=
707          sprintf ("(Results are correct with %.02f percent confidence.\n", $correct1*100) .
708          sprintf (" Number of valid samples must be higher than %d to get results with %.02f percent
     correctness.)\n", $c_n2, $correct2*100);
709  }
710  else {
711      $result_string .=
712                  "(Warning: Number of valid samples is $samples_considered,\n" .
713          sprintf (" but must be higher than %d to get results with %.02f percent confidence\n", $c_n1
     , $correct1*100) .
714          sprintf (" and must be higher than %d to get results with %.02f percent confidence.)\n",
     $c_n2, $correct2*100);
715  }
716
717  if ($key eq 1) {
718      $result_string .= &print_key();
719  }
720  print $result_string;
721
722  # END: RESULT STRING FOR EOS ######################################################
723
724
725  # RESULT TO FILE ##################################################################
726  print RESULT $result_string;
727  close RESULT;
728  # END: RESULT TO FILE##############################################################
729
730
731
732  # DATABASES - DISCONNECT ##########################################################
733  $dbh_mysql->disconnect;
734  $dbh_oracle->disconnect;
735  # END: DATABASES - DISCONNECT #####################################################
736
737
738
739  # SUBROUTINES #####################################################################
740  sub help {
741
742      print STDERR "
743  EOS - Analysing Nyx based on LRZ Netzdokumentation (2008)
744
745  No warranty of any kind!
746
747  ========================================
748
749  Usage: <perl> $0
750
751  mandatory:
752   -samples=<int>: maximum number of samples to test (default: 1000)
753   -s=<int>: (same as --samples=<int>)
```

```
754
755   -dataport_loading=<float [0...1]>: loading
756   -dl=<float [0...1]>: (same as --dataport_loading=<float [0...1]>)
757
758
759  optional:
760   -heuristics: consider 'auto_updownlink_override'
761   -h: (same as --heuristics)
762
763   -accesspoints: consider accesspoints as updownlink
764   -a: (same as --ap)
765
766   -key: print additional key for used abbreviations
767
768   -keyonly: print only key for used abbreviations
769
770   -help: (shows this message)
771  ";
772
773      exit 0 ;
774
775  }
776
777  sub print_key() {
778
779      my $key_string = "
780  ------------------------------------------------------------------------------|
781  Key:                                                                          |
782   NTP = number of updownlinkports                                              |
783   NFP = number of false positives                                              |
784   NFN = number of false negatives                                              |
785  aNTN = number of dataports minus adjustement value calculated by dl           |
786         where dl is a dataport loading as adjustment for residual calculation  |
787         = NTN - (NTN * dl)                                                     |
788    aV = adujusted Value = NTN * dl                                             |
789                                                                                |
790                                                  Actual value                  |
791                                        updownlink (positiv) dataport (negativ) |
792                                                                                |
793                      updownlink (positiv)         NTP              NFP        |
794  Prediction                                                                    |
795                        dataport (negativ)         NFN       aNTN+adjustmentValue |
796                                                                                |
797                                                                                |
798  True Positive ratio : TP =  NTP/(NTP+NFN)                                     |
799  True Negative ratio : TN = aNTN/(NFP+NTN)                                     |
800  False Positive ratio: FP =  NFP/(NFP+NTN)                                     |
801  False Negative ratio: FN =  NFN/(NTP+NFN)                                     |
802  AdjustmentError     : AE =   aV/(NFP+NTN)                                     |
803                                                                                |
804  ACcuracy:                                                                     |
805   dataports not considered: AC = TP = NTP/(NTP+NFP)                            |
806   dataports considered    : AC = (NTP+aNTN)/(NTP+NFP+NFN+aNTN+AE)               |
807  ------------------------------------------------------------------------------|
808  ";
809
810      return $key_string;
811
812  }
813
814  # adapted from: http://www.hidemail.de/blog/datum-uhrtime-perl.shtml
815  sub get_time {
816      #my @wochentage = ("Montag","Dienstag","Mittwoch","Donnerstag","Freitag","Samstag","Sonntag");
817      #my @monatsnamen = ("","Januar","Februar","März","April",
     "Mai","Juni","Juli","August","September","Oktober", "November","Dezember");
818      my ($sec,$min,$hour,$heutetag,$heutemonat,$heutejahr,$wday,$yday,$isdst) = localtime(time);
819      $heutemonat++;
820      $heutejahr+=1900;
821      $wday--;
822      if ($wday eq '-1'){$wday=6;}
823
824      $hour = sprintf "%02d",$hour;
825      $min = sprintf "%02d",$min;
826      $sec = sprintf "%02d",$sec;
827
828      my %werte = (
829          hour => $hour,
830          minute => $min,
831          #sekunde => $sec,
832          day => $heutetag,
```

```
833             month => $heutemonat,
834             year => $heutejahr,
835             #wochentag => $wday,
836             #jahrestag => $yday,
837             #sommerzeit => $isdst,
838             #wochentagtext => $wochentage[$wday],
839             #monattext => $monatsnamen[$heutemonat],
840         );
841
842         return %werte;
843     }
844
845     sub progress {
846         my $backspaces = $_[0]; # first parameter
847         my $progress   = $_[1]; # second parameter
848         my $digits = length($progress);
849
850         print $backspaces;
851         $backspaces = "\b" x $digits;
852         print STDOUT $progress;
853         $progress++;
854         return ($progress, $backspaces);
855     }
856
857     # END: SUBROUTINES ###################################################################
858
```

D Analyse: LRZ Rechnerwürfel

Im Folgenden sind die genauen Ergebnisse für die Überprüfung von 124 Ports aus dem LRZ Rechnerwürfel im Detail aufgeführt. Es wurden mehr als 100 Ports kontrolliert, weil im Vorhinein nicht erkennbar ist, ob ein Port tatsächlich vorhanden und aktiv ist. Man sieht dies auch daran, dass von den 124 überprüften Ports nur 102 gültig sind. Die Zahl von 100 zu testenden Ports ist ein Kompromiss aus Repräsentativität und Aufwand, der durch die manuelle Sichtung entsteht.

Die Spalte *UPDOWNLINK* bezeichnet die Klassifikation durch den maschinellen Lernalgorithmus. Die Spalte *SICHTUNG* die Klassifikation durch die manuelle Sichtung. Dabei steht der Wert „1“ für einen Up- bzw. Downlinkport und der Wert „0“ für einen Datenport. Das Minus-Zeichen „-“ beschreibt einen nicht überprüfbaren Port. Die Gründe dafür können unterschiedlich sein. Beispielsweise kann der Port mit der Portnummer nicht vorhanden sein oder der Port ist nicht aktiv. In diesen Fällen ist eine Klassifikation durch Sichtung und eine Beurteilung der Nyx-Klassifikation nicht möglich.

Ein Port ist genau dann korrekt erkannt, wenn beide Werte, *UPDOWNLINK* und *SICHTUNG*, übereinstimmen.

Das Ergebnis der Überprüfung ist eine Erkennungsrate (accuracy) von 99%.

DEVICE	IFINDEX	IFDESCRIPTION	IFSPEED	UPDOWNLINK	SYSNAME	SYSLOCATION	SICHTUNG
HPJ4899ATW432MZ0DS	39	39	100 Megabit/s	0	SWI1-1WR	DAR Rack 5E Mgmt der TSM-Server	0
HPJ4899ATW432MZ0DS	32	32	10 Megabit/s	0	SWI1-1WR	DAR Rack 5E Mgmt der TSM-Server	0
HPJ4904ASG547SK06D	14	14	1,0 Gigabit/s	0	SWK1-1WR	DAR Rack 5J	-
HPJ4904ASG547SK06D	46	46	10 Megabit/s	0	SWK1-1WR	DAR Rack 5J	-
HPJ4906ASG544SG03N	12	12	1,0 Gigabit/s	0	SWM2-1WR	DAR Rack 5L	0
HPJ4906ASG544SG03N	45	45	1,0 Gigabit/s	0	SWM2-1WR	DAR Rack 5L	0
HPJ4906ASG544SG03N	11	11	1,0 Gigabit/s	0	SWM2-1WR	DAR Rack 5L	0
HPJ4906ASG544SG03N	29	29	1,0 Gigabit/s	0	SWM2-1WR	DAR Rack 5L	0
HPJ4906ASG544SG03N	10	10	10 Megabit/s	0	SWM2-1WR	DAR Rack 5L	0
HPJ4903ASG519SJ00W	13	13	1,0 Gigabit/s	0	SWK6-3WR	HRR Rack 10A	0
HPJ4904ASG605SK06V	12	12	100 Megabit/s	0	SWK2-3WR	HRR Reihe 1 Metadatenserver	0
HPJ4904ASG605SK06V	44	44	10 Megabit/s	0	SWK2-3WR	HRR Reihe 1 Metadatenserver	0
HPJ4904ASG605SK06V	39	39	1,0 Gigabit/s	0	SWK2-3WR	HRR Reihe 1 Metadatenserver	0
HPJ4904ASG605SK078	16	16	100 Megabit/s	0	SWK1-3WR	HRR Reihe 1 Metadatenserver	0
HPJ4904ASG605SK05W	31	31	100 Megabit/s	0	SWK4-3WR	HRR Reihe 2 Mitte	0
HPJ4904ASG706SK05J	13	13	100 Megabit/s	0	SWK7-3WR	HRR Reihe 2 Mitte rechts	0
HPJ4906ASG552SG07N	45	45	10 Megabit/s	0	SWM1-2WR	LRZ-NSR Reihe 8F	-
HPJ8698ASG702SV009	61	C13	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	70	C22	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	82	D10	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	85	D13	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	88	D16	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	91	D19	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	115	E19	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	118	E22	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG702SV009	119	E23	1,0 Gigabit/s	1	SWZ4-2WR	LRZ-Rechnerwuerfel NSR 8D 1 Zentralswitch	1
HPJ8698ASG652SV00G	61	C13	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	62	C14	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	65	C17	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	82	D10	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	85	D13	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	88	D16	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	91	D19	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	93	D21	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	95	D23	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	96	D24	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	109	E13	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ8698ASG652SV00G	119	E23	1,0 Gigabit/s	1	SWZ5-2WR	LRZ-Rechnerwuerfel NSR 8D 2 Zentralswitch	1
HPJ4904ASG547SK066	39	39	100 Megabit/s	0	SWK11-2WR	NSR Rack 6J Solaris und Linux-Server	0
HPJ9050ASG739KJ04J	31	31	1,0 Gigabit/s	0	SWO1-2WR	NSR 1 Linux-Cluster	-
HPJ4899BCN701SD0K9	11	11	100 Megabit/s	0	SWI8-2WR	NSR 1S	0
HPJ4899BCN701SD1A5	34	34	100 Megabit/s	0	SWI9-2WR	NSR 1U	0

DEVICE	IFINDEX	IFDESCRIPTION	IFSPEED	UPDOWNLINK	SYSNAME	SYSLOCATION	SICHTUNG
HPJ4906ASG652SG01U	29	29	10 Megabit/s	0	SWM12-2WR	NSR 1X Linux-Cluster	0
HPJ4906ASG442SG055	24	24	1,0 Gigabit/s	0	SWO20-2WR	NSR 2I Linux-Cluster	0
HPJ4899BCN701SD0W8	17	17	100 Megabit/s	0	SWI10-2WR	NSR 2K Linux-Cluster	0
HPJ4906ASG526SG02A	12	12	100 Megabit/s	0	SWM2-2WR	NSR 2W HLRB II Visualisierungssystem	0
HPJ4899BCN731SD05P	27	27	100 Megabit/s	0	SWI17-2WR	NSR 3E Linux-Cluster	0
HPJ4904ASG618SK09Y	39	39	10 Megabit/s	0	SWK34-2WR	NSR 6B	0
HPJ4904ASG618SK09Y	34	34	1,0 Gigabit/s	0	SWK34-2WR	NSR 6B	0
HPJ4904ASG618SK09Y	40	40	10 Megabit/s	0	SWK34-2WR	NSR 6B	0
HPJ4904ASG618SK09Y	22	22	1,0 Gigabit/s	0	SWK34-2WR	NSR 6B	0
HPJ4904ASG618SK09Y	10	10	1,0 Gigabit/s	0	SWK34-2WR	NSR 6B	0
HPJ9022ACN637XJ0DG	17	17	10 Megabit/s	0	SWN1-2WR	NSR 6G Housing von Fremdservern	0
HPJ9022ACN637XJ0DG	23	23	10 Megabit/s	0	SWN1-2WR	NSR 6G Housing von Fremdservern	0
HPJ4899BCN701SD1A1	47	47	100 Megabit/s	0	SWI11-2WR	NSR 6G Iom-Anschluesse	0
HPJ4899BCN701SD1A1	46	46	100 Megabit/s	0	SWI11-2WR	NSR 6G Iom-Anschluesse	0
HPJ4899BCN701SD0PD	20	20	100 Megabit/s	0	SWI12-2WR	NSR 7G Iom-Anschluesse	0
HPJ9050ASG738KJ06X	19	19	1,0 Gigabit/s	1	SWO12-2WR	NSR 7G VMware-Bereich	-
HPJ4903ASG618SJ0H2	25	25	10 Megabit/s	0	SWO12-2WR	NSR 7G VMware-Bereich	-
HPJ4903ASG618SJ0H2	20	20	1,0 Gigabit/s	0	SWO12-2WR	NSR 7G VMware-Bereich	-
HPJ9050ASG734KJ06F	43	43	10 Megabit/s	0	SWO13-2WR	NSR 7G VMWare-Bereich	-
HPJ9050ASG734KJ06F	38	38	10 Megabit/s	0	SWO13-2WR	NSR 7G VMWare-Bereich	-
HPJ9050ASG734KJ06F	17	17	1,0 Gigabit/s	1	SWO13-2WR	NSR 7G VMWare-Bereich	-
HPJ9050ASG734KJ06F	47	47	10 Megabit/s	1	SWO13-2WR	NSR 7G VMWare-Bereich	-
HPJ4904ASG437SK015	31	31	10 Megabit/s	0	SWK19-2WR	NSR 7V	0
HPJ4904ASG437SK015	13	13	10 Megabit/s	0	SWK19-2WR	NSR 7V	0
HPJ4906ASG641SG029	18	18	1,0 Gigabit/s	0	SWM10-2WR	NSR Rack 1I	0
HPJ4899BCN731SD05E	20	20	100 Megabit/s	0	SWI16-2WR	NSR Rack 1N Linux-Cluster-Bereich	0
HPJ4899BCN731SD05E	14	14	100 Megabit/s	0	SWI16-2WR	NSR Rack 1N Linux-Cluster-Bereich	0
HPJ4899BCN731SD0FX	50	50	1,0 Gigabit/s	1	SWI15-2WR	NSR Rack 1O Linux-Cluster-Bereich	1
HPJ4899BCN731SD0FX	35	35	10 Megabit/s	0	SWI15-2WR	NSR Rack 1O Linux-Cluster-Bereich	0
HPJ4899BCN731SD0DK	29	29	100 Megabit/s	0	SWI14-2WR	NSR Rack 1Q Linux-Cluster-Bereich	0
HPJ4899BCN731SD0D0	50	50	1,0 Gigabit/s	1	SWI13-2WR	NSR Rack 1R Linux-Cluster-Bereich	1
HPJ4899BCN731SD0D0	27	27	100 Megabit/s	0	SWI13-2WR	NSR Rack 1R Linux-Cluster-Bereich	0
HPJ4906ASG649SG005	35	35	10 Megabit/s	0	SWM11-2WR	NSR Rack 1S Linux-Cluster	0
HPJ4906ASG542SG02E	12	12	1,0 Gigabit/s	0	SWM9-2WR	NSR Rack 1U Linux-Cluster	0
HPJ4906ASG438SG00C	19	19	10 Megabit/s	0	SWM3-2WR	NSR Rack 1Z Linux-Cluster	-
HPJ4906ASG438SG00C	40	40	100 Megabit/s	0	SWM3-2WR	NSR Rack 1Z Linux-Cluster	-
HPJ8698ASG640SV02S	220	J4	4,0 Gigabit/s	1	SWZ3-2WR	NSR Rack 2A Zentralswitch Linux-Cluster	1
HPJ8698ASG640SV02S	4	A4	4,0 Gigabit/s	1	SWZ3-2WR	NSR Rack 2A Zentralswitch Linux-Cluster	0
HPJ8698ASG640SV02S	243	K3	4,0 Gigabit/s	0	SWZ3-2WR	NSR Rack 2A Zentralswitch Linux-Cluster	0
HPJ4813ASG405NV335	11	11	10 Megabit/s	0	SWL1-2WR	NSR Rack 2E Management (Linux-Cluster)	0
HPJ4906ASG438SG014	38	38	1,0 Gigabit/s	0	SWM5-2WR	NSR Rack 2F Linux-Cluster	0
HPJ4906ASG438SG014	22	22	10 Megabit/s	0	SWM5-2WR	NSR Rack 2F Linux-Cluster	0
HPJ4906ASG438SG02P	16	16	1,0 Gigabit/s	0	SWO19-2WR	NSR Rack 2H Linux-Cluster	-

DEVICE	IFINDEX	IFDESCRIPTION	IFSPEED	UPDOWNLINK	SYSNAME	SYSLOCATION	SICHTUNG
HPJ4906ASG438SG02P	51	A3	4,0 Gigabit/s	0	SWO19-2WR	NSR Rack 2H Linux-Cluster	-
HPJ4865ASG30802590	53	C5	1,0 Gigabit/s	0	SWK41-2WR	NSR Rack 5E BVB-Bereich	-
HPJ4865ASG30802590	125	F5	1,0 Gigabit/s	0	SWK41-2WR	NSR Rack 5E BVB-Bereich	-
HPJ9021ACN644XI0Y9	13	13	10 Megabit/s	0	SWN6-2WR	NSR Rack 5F BVB-Bereich	0
HPJ9021ACN644XI066	10	10	10 Megabit/s	0	SWN8-2WR	NSR Rack 5H BVB-Bereich	0
HPJ9021ACN644XI0YM	22	22	1,0 Gigabit/s	0	SWN9-2WR	NSR Rack 5I BVB-Bereich	0
HPJ9022ACN644XJ0GU	46	46	10 Megabit/s	0	SWN10-2WR	NSR Rack 5K BVB-Bereich	0
HPJ9022ACN644XJ0GU	15	15	100 Megabit/s	0	SWN10-2WR	NSR Rack 5K BVB-Bereich	0
HPJ4899BCN701SD0KK	15	15	100 Megabit/s	0	SWI6-2WR	NSR Rack 6B	0
HPJ4903ASG546SJ0EZ	15	15	10 Megabit/s	0	SWK22-2WR	NSR Rack 6E	0
HPJ4903ASG548SJ06U	24	24	1,0 Gigabit/s	1	SWK23-2WR	NSR Rack 6F	1
HPJ4903ASG618SJ02H	18	18	1,0 Gigabit/s	0	SWK36-2WR	NSR Rack 6Y Windowsbereich	0
HPJ4899BCN701SD1AI	49	49	10 Megabit/s	0	SWI5-2WR	NSR Rack 7B	0
HPJ4899BCN701SD1AI	26	26	10 Megabit/s	0	SWI5-2WR	NSR Rack 7B	0
HPJ4903ASG546SJ0EX	18	18	10 Megabit/s	0	SWK28-2WR	NSR Rack 7E	0
HPJ4903ASG546SJ0CB	18	18	100 Megabit/s	0	SWK30-2WR	NSR Rack 7F	0
HPJ4903ASG546SJ0CB	19	19	1,0 Gigabit/s	0	SWK30-2WR	NSR Rack 7F	0
HPJ4904ASG636SK07K	32	32	10 Megabit/s	0	SWK16-2WR	NSR Rack 7J Solaris, Linux-Server	0
HPJ4904ASG636SK07K	13	13	10 Megabit/s	0	SWK16-2WR	NSR Rack 7J Solaris, Linux-Server	0
HPJ4904ASG636SK07K	11	11	100 Megabit/s	0	SWK16-2WR	NSR Rack 7J Solaris, Linux-Server	0
HPJ4903ASG546SJ0DL	51	A3	4,0 Gigabit/s	1	SWO17-2WR	NSR Rack 7T Anbindung der NASFiler	-
HPJ4903ASG712SJ0I7	24	24	1,0 Gigabit/s	1	SWK37-2WR	NSR Rack 7U	1
HPJ4903ASG712SJ0I7	15	15	10 Megabit/s	0	SWK37-2WR	NSR Rack 7U	0
HPJ4903ASG548SJ06X	13	13	10 Megabit/s	0	SWK8-2WR	NSR Rack 7U AFS-Server	0
HPJ4904ASG437SK01C	26	26	10 Megabit/s	0	SWK18-2WR	NSR Rack 7V Fileserver	0
HPJ4903ASG603SJ04B	11	11	1,0 Gigabit/s	0	SWK32-2WR	NSR Rack 7Y	0
HPJ4899BTW547SD1U6	13	13	10 Megabit/s	0	SWI4-2WR	NSR Rack 8E Sammelswitch	-
HPJ4899BTW536SD003	10	10	100 Megabit/s	0	SWI1-2WR	NSR Rack 8G 1 Switch im Facilitynetz	0
HPJ4899BTW536SD05S	20	20	10 Megabit/s	0	SWI3-2WR	NSR Rack 8G 3 Switch im Facilitynetz	0
HPJ4899BTW536SD05S	38	38	100 Megabit/s	0	SWI3-2WR	NSR Rack 8G 3 Switch im Facilitynetz	0
HPJ8697ASG611SU00T	14	A14	10 Megabit/s	0	SWZ1-2WR	NSR Rack 8G Telefonie	0
HPJ8697ASG611SU00T	5	A5	100 Megabit/s	0	SWZ1-2WR	NSR Rack 8G Telefonie	0
HPJ8697ASG605SY04P	18	A18	10 Megabit/s	0	SWZ1-2WR	NSR Rack 8G Telefonie	0
HPJ4903ASG546SJ0EV	18	18	100 Megabit/s	0	SWK12-2WR	NSR, Rack 6K, Solaris+Linux-Server	0
HPJ4903ASG546SJ0DV	23	23	1,0 Gigabit/s	1	SWK2-2WR	SLB-Switch 2 Reihe 6E Linuxber	1
HPJ4903ASG548SJ07A	15	15	100 Megabit/s	0	SWK3-2WR	SLB-Switch 3 Reihe 7Q Solaris	0

Gültige Ports	102	**Ergebnis**	**99%**
Korrekt erkannte Ports	101	(Accuracy)	
Falsch erkannte Ports	1		