FAKULTÄT FÜR !NFORMATIK

# Konzeption und prototypische Umsetzung von Network Discovery-Verfahren bei minimalen Vorkenntnissen über die Systemlandschaft

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering/Internet Computing**

eingereicht von

**Thomas Terényi**
Matrikelnummer 0225440

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Thomas Grechenig
Mitwirkung: Gerald Fischer

Wien, 24.07.2009

(Unterschrift Verfasser) (Unterschrift Betreuer)

# Konzeption und prototypische Umsetzung von Network Discovery-Verfahren bei minimalen Vorkenntnissen über die Systemlandschaft

## DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering/Internet Computing**

eingereicht von

**Thomas Terényi**

Matrikelnummer 0225440

ausgeführt am

Institut für Rechnergestützte Automation

Forschungsgruppe Industrial Software

der Fakultät für Informatik der Technischen Universität Wien

**Betreuung:**

Betreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Thomas Grechenig

Mitwirkung: Gerald Fischer

Wien, 24.07.2009

# Erklärung

**Thomas Terényi, Gassergasse 29/15, 1050 Wien**

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 24.07.2009

_____

(Thomas Terényi)

# Kurzfassung

Diese Arbeit behandelt das Thema *Network Discovery* – das Auffinden von Geräten in einem Computernetzwerk und von Informationen sowohl über diese selbst, als auch über deren Verbindungen untereinander. Network Discovery fällt in den Bereich des Network Managements und kann als Basis für viele der dazugehörigen Tätigkeiten dienen. Diese Arbeit beschreibt nicht nur die praktische Relevanz des Themas, sondern liefert auch einerseits eine Einführung in den Themenkomplex mit seinen wichtigsten Aspekten und behandelt andererseits konkrete Discovery-Techniken im Detail. Einige der vorgestellten Methoden werden in einem Algorithmus zusammengefasst, wobei besonderes Augenmerk auf ein Discovery bei minimalen Vorkenntnissen (d.h. a priori wird möglichst wenig Wissen über das Netzwerk benötigt) gelegt wird. Weiters wird dieser Algorithmus in dem Programm *Network Explorer* umgesetzt, das eigens im Rahmen dieser Arbeit entwickelt wurde. Dieses Programm wird in einer realen Umgebung zum Einsatz gebracht und dessen Ergebnisse werden vorgestellt, analysiert und bewertet. Anschließend werden konkrete, mögliche Szenarien für dessen praktische Anwendung beschrieben. Die wichtigsten Ziele dieser Arbeit sind eine Einführung in das Thema Network Discovery zu liefern sowie eine Überblick über Methoden und Technologien in diesem Bereich zu geben. Die vorgestellten Techniken werden vergleichend analysiert und ein Leitfaden zur deren Auswahl bei der Entwicklung eines Discovery-Systems erstellt. Anhand eines konkreten Algorithmus' und dessen Implementierung in einem Programm wird beispielhaft demonstriert, wie ein derartiges System erstellt werden kann. Weiters wird gezeigt, dass dieses System auch praxistauglich ist und eine Reihe von realen Anwendungen besitzt. Bei allen in dieser Arbeit vorgestellten Aspekten steht besonders der praktische Nutzen im Vordergrund.

# Abstract

This thesis covers the subject of *Network Discovery* – finding devices in a computer network, as well as information about them and their connections. Network Discovery is a part of the more general area of Network Management and can serve as the basis for many tasks in this field. This thesis does not only highlight the practical relevance of the subject, but also provides an introduction to a whole range of related topics. It also covers specific discovery methods in detail. Some of the presented techniques are selected and integrated into a single discovery algorithm. A core aspect of this algorithm is a discovery with minimum previous knowledge (i.e. as little information as possible about the network is necessary). This algorithm is then implemented in a program called *Network Explorer*, which has been developed as part of this thesis. The program is employed in an actual corporate network and its results are presented, analyzed, and evaluated. Furthermore, examples and scenarios for the use of this program are addressed. The main goals of this thesis are: to provide an introduction to this subject; to list, explain, and compare concrete discovery techniques resulting in a guide for selecting methods when developing a discovery system; and to give an example of how to do so by presenting and describing a specific algorithm and a program implementing this algorithm. In addition, it is shown that this system provides solutions to real-world problems. The practical usability of the concepts described herein is the main focus throughout this thesis.

# Danksagungen

Ich möchte meinen Eltern für ihre stete Unterstützung, besonders meinem Vater für das Korrekturlesen dieser Arbeit sowie der Firma *Novartis* für die Erlaubnis, das Programm *Network Explorer* in ihrem Netzwerk testen zu dürfen, danken.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **ACK** | Acknowledgement |
| **AFT** | Address Forwarding Table |
| **API** | Application Programming Interface |
| **ARP** | Address Resolution Protocol |
| **ASCII** | American Standard Code for Information Interchange |
| **ASN.1** | Abstract Syntax Notation One |
| **ATM** | Asynchronous Transfer Mode |
| **BGP** | Border Gateway Protocol |
| **CDP** | Cisco Discovery Protocol |
| **CIFS** | Common Internet File System |
| **CSMA/CD** | Carrier Sense Multiple Access/Collision Detection |
| **CSV** | Comma-Separated Values |
| **DNS** | Domain Name System |
| **EBCDIC** | Extended Binary Coded Decimals Interchange Code |
| **EGP** | Exterior Gateway Protokoll |
| **EN** | Europäische Norm |
| **FCAPS** | Fault, Configuration, Accounting, Performance, Security |

**FDB** Forwarding Database

**FDDI** Fiber Distributed Data Interface

**FDM** Frequency Division Multiplexing

**FIN** Finish

**FTP** File Transfer Protocol

**HP** Hewlett-Packard

**HSRP** Hot Standby Router Protocol

**HTTP** Hypertext Transfer Protocol

**ICMP** Internet Control Message Protocol

**IEEE** Institute of Electrical and Electronics Engineers

**IETF** Internet Engineering Task Force

**IGRP** Interior Gateway Routing Protocol

**IOS** Internetwork Operating System

**IP** Internet Protocol

**IPX** Internetwork Packet Exchange

**ISDN** Integrated Services Digital Network

**ISO** International Organization for Standardization

**ITU** International Telecommunication Union

**LAG** Link Aggregation

**LAN** Local Area Network

**LLDP** Link Layer Discovery Protocol

**MAC** Media Access Control

| | |
|---|---|
| **MIB** | Management Information Base |
| **NAT** | Network Address Translation |
| **NBNS** | NetBIOS Naming Service |
| **NBT** | NetBIOS over TCP/IP |
| **NCL** | Network Client Locator |
| **NetBIOS** | Network Basic Input Output System |
| **NIBR** | Novartis Institutes for Biomedical Research |
| **NIC** | Network Interface Controller |
| **OAM** | Operation, Administration and Maintenance |
| **OID** | Object Identifier |
| **OS** | Operating System |
| **OSI** | Open Systems Interconnection |
| **OSPF** | Open Shortest Path First |
| **OUI** | Organizationally Unique Identifier |
| **RFC** | Request for Comments |
| **RIP** | Routing Information Protocol |
| **RMI** | Remote Method Invocation |
| **RST** | Reset |
| **SMB** | Server Message Block |
| **SMTP** | Simple Mail Transfer Protocol |
| **SNMP** | Simple Network Management Protocol |
| **SPX** | Sequenced Packet Exchange |

| | |
|---|---|
| **SQL** | Structured Query Language |
| **SSDP** | Simple Service Discovery Protocol |
| **SSH** | Secure Shell |
| **STP** | Spanning Tree Protocol |
| **SYN** | Synchronization |
| **TCP** | Transmission Control Protocol |
| **TDM** | Time Division Multiplexing |
| **TTL** | Time to Live |
| **UDP** | User Datagram Protocol |
| **UMTS** | Universal Mobile Telecommunications System |
| **UPnP** | Universal Plug and Play |
| **VLAN** | Virtual Local Area Network |
| **VM** | Virtual Machine |
| **VRRP** | Virtual Router Redundancy Protocol |
| **VTP** | VLAN Trunking Protocol |
| **WAN** | Wide Area Network |
| **WBEM** | Web Based Enterprise Management |
| **WLAN** | Wireless Local Area Network |
| **WMI** | Windows Management Instrumentation |

# Kapitel 1

# Einleitung

Dieses Kapitel beschreibt die Problemstellung, mit der sich diese Arbeit auseinandersetzt, die Motivation diese zu behandeln, das konkrete Ziel der Arbeit sowie explizite Nicht-Ziele. Weiters wird der aktuelle Stand der Forschung auf diesem Gebiet betrachtet sowie eine Übersicht über den Inhalt dieser Arbeit geliefert.

## 1.1 Problemstellung

Computernetzwerke sind heutzutage ein nicht mehr wegzudenkender Teil unseres alltäglichen Lebens, auch wenn sie ihre Aufgaben meist diskret im Hintergrund, unbemerkt und unbeachtet von den Benutzern, verrichten (zumindest solange bis einmal etwas nicht funktioniert). Bestanden Netzwerke in früher Zeit aus großen und teuren Mainframes mit vielen angeschlossenen Terminals, so sind heute bereits Telefone, Fernseher oder gar Küchengeräte vernetzt. Der ideale Mensch des 21. Jahrhunderts ist immer online, egal ob zuhause, im Büro oder unterwegs – so wird es uns zumindest glauben gemacht (siehe z.B. [68]).

Dass wir die Vorzüge der Vernetzung nützen können, ist aber keine Selbstverständlichkeit, sondern Computernetzwerke müssen verwaltet, betreut und gewartet werden, wobei die ständig steigende Anzahl an verbundenen Geräten sowie deren immer größer werdende Heterogenität diese Aufgabe nicht unbedingt erleichtern. Zur Administration von Netzwerken (*Network Management*[1]) – in dieser Arbeit liegt der Fokus besonders auf Firmen- oder Universitätsnetzwerken und dergleichen – gehören verschiedenste Aufgaben, unter anderem zum Beispiel die Inventarisierung von Geräten (*Asset*

[1] Diese sowie weitere in dieser Einführung vorkommende Fachbegriffe werden im Laufe dieser Arbeit (siehe Kapitel *2 Einführung in Computernetzwerke* und Kapitel *3 Grundlagen des Network Discoverys*) noch genau definiert und näher betrachtet.

*Management*), deren Überwachung im Betrieb (*Network Monitoring*) oder die Verwaltung von deren Einstellungen (*Configuration Management*).

Durch die hohe Anzahl von Geräten in heutigen Netzwerken (tausende in einem typischen Firmennetz) und dadurch, dass diese leicht von Jedermann an das Netzwerk angeschlossen bzw. von diesem getrennt werden können, ist es schwierig, stets eine genaue Übersicht über das Netzwerk zu behalten und nahezu unmöglich, händisch eine Netzwerkdokumentation ständig aktuell zu halten. Daher liegt es nahe zu versuchen, Aufgaben aus diesem Bereich zu automatisieren bzw. technisch zu unterstützen.

Diese Arbeit beschäftigt sich mit einem Teilaspekt des Network Managements, dem *Network Discovery*. Dessen Aufgabe ist es, möglichst alle Geräte in einem Netzwerk aufzuspüren sowie gleichzeitig Informationen über diese sowie über deren Verbindungen untereinander (die Topologie) zu bestimmen. Diese Informationen können als Basis für alle anderen Aufgaben des Network Managements verwendet werden. Bevor Geräte überwacht werden können, muss etwa zuerst feststehen, welche Geräte es überhaupt zum Überwachen gibt. Genauso können die Daten aus dem Network Discovery als Ausgangspunkt für die Inventarisierung dienen oder zusätzliche Informationen für die Konfigurationsverwaltung liefern.

Obwohl ein Netzwerk implizit „weiß“, welche Geräte wie und wo angeschlossen sind – die Kommunikation mit ihnen funktioniert meist auf Anhieb und die Strom- oder Lichtimpulse bzw. Funksignale finden geradezu „magisch“ den richtigen Weg durch das Netz, manchmal sogar über ganze Kontinente hinweg –, so ist es dennoch kein Leichtes, diese Informationen auch explizit zu machen, sondern komplexe Vorgehensweisen sind nötig.

Auch die Relevanz der Information, wo sich ein bestimmtes Gerät physisch befindet, ist nicht zu vernachlässigen. Nicht selten haben sich Support-Mitarbeiter in einer Situation wiedergefunden, wo sie nach einem bestimmten Gerät gesucht haben (sei um es gegen ein neues auszutauschen oder weil ein Drucker etwa einen Papierstau gemeldet hat), nur um es verwundert nicht dort vorzufinden, wo es laut Aufzeichnungen sein sollte (in einem Netz mit tausenden Geräten hat niemand mehr im Kopf, was genau wo ist). Oft stellt es sich dann erst nach langem Suchen heraus, dass irgendjemand es irgendwann einmal ab- oder umgesteckt hat, ohne irgendwen davon in Kenntnis zu setzen. Das Problem hierbei ist, dass die Geräte oft weiter normal funktionieren, auch wenn sie z.B. übersiedelt werden (heutige Netzwerke sind „schlau“ genug, um mit solchen Situationen umgehen zu können, so sie nicht aus Sicherheitsgründen explizit derart konfiguriert sind, die Funktion in diesem Fall zu verweigern), dies also anfänglich niemandem auffällt.

Weitere typische Szenarien (vgl. auch Kapitel *7 Einsatzszenarien*) im Alltag eines Netzwerkmanagers sind zum Beispiel das Finden von Geräten, die Benutzer unerlaubt an das Netzwerk angeschlossen haben, die zuvor genannte Standortbestimmung sowie das Verfolgen von Bewegungen von Geräten, das Erkennen von Konfigurationsfehlern, der Abgleich von dynamischen Netzwerkdaten (z.B. IP-Adressen) mit statischen Asset-Daten (z.B. Seriennummern) und viele mehr.

In solchen Situationen wäre es hilfreich, diese Aufgaben entweder komplett automatisch oder zumindest mit technischer Unterstützung durchführen zu können (der Techniker sieht z.B. einfach auf seinem Computer nach, wo sich ein Gerät befindet, anstatt es langwierig zu suchen). Weiters lassen sich alle der genannten Fälle auf das Problem des Auffindens von Geräten und Informationen über diese – also das Network Discovery – zurückführen.

Wie die angeführten Beispiele zeigen, ist das Thema des Network Discoverys einerseits praktisch relevant, aber andererseits auch theoretisch und akademisch interessant, da Computernetzwerke komplexe Systeme sind und das Extrahieren der gewünschten Informationen oft nicht einfach, sondern im Gegenteil äußerst diffizil ist (vgl. Kapitel *4 Der Discovery-Algorithmus*).

## 1.2 Motivation

Die Motivation zur Behandlung dieses Themas lässt sich grob in dem Satz zusammenfassen, dass dieses Thema zwar einerseits – wie im vorigen Abschnitt gezeigt – relevant ist, andererseits aber bisherige Ansätze (vgl. Abschnitt *1.3 Stand der Forschung*) teilweise unzulänglich sind.

Folgende Punkte beschreiben einige Aspekte der Motivation im Detail:

- Bisherige Arbeiten beschäftigen sich meist jeweils immer nur mit einem isolierten Teilaspekt des Network Discoverys.
- Es existiert keine umfassende Einführung in die gesamte Thematik mit all ihren Aspekten.
- Oft bearbeiten Papers ihr Thema entweder ausschließlich rein theoretisch oder setzen nur genau den einen behandelten Aspekt auch in einem Programm um.
- In Arbeiten (z.B. [82]), bei denen versucht wird, mehrere Techniken in einer Software zusammenzufassen, ist der Ansatz häufig der, bereits bestehende Tools zusammenzuschließen. Es wird aber nicht gezeigt, wie von Grund auf ein zusammenhängender Algorithmus entworfen und ein komplettes System gebaut werden kann.

- Viele Arbeiten (z.B. [21]), die als Thema das Erkennen der Netzwerk-Topologie haben, beschäftigen sich mit der (Layer 3) Topologie des gesamten Internets, nicht aber mit LAN-Topologien (auf Layer 2 und Layer 3).
- In bestehenden Papers werden Technologien, die heutzutage Teil eines praktisch jeden Firmen-LANs sind (z.B. VLANs, Link Aggregation, virtuelle Router, virtuelle Maschinen etc.) gar nicht oder nur kaum berücksichtigt.
- Generell beschäftigt sich bestehende Literatur zu diesem Thema (z.B. [45], [10], [5] und [78]) vor allem mit akademischen Korrektheitsbeweisen in konstruierten Fällen für diverse Algorithmen, die aber allesamt nicht praxistauglich sind, da dafür Annahmen getroffen werden und Einschränkungen gelten müssen, die so in einer realen Umgebung nicht haltbar sind. Dies hängt auch damit zusammen, dass die zuvor genannten Technologien, die heute überall verwendet werden, dort nicht berücksichtigt werden.
- Bestehende kommerzielle Software beschäftigt sich hauptsächlich mit Network Monitoring (wo die Administratoren die vorhandenen Geräte und das Netz kennen müssen) und meist nur nebensächlich oder gar nicht mit Network Discovery.
- Vorhandene Software-Tools beschäftigen sich meist mit dem Finden von Geräten *oder* dem Erkennen der Topologie, nicht aber mit beidem (wobei sich dies ideal ergänzen würde und Ergebnisse aus einem Teil für den anderen verwendet werden könnten und umgekehrt, anstatt dass die fehlenden Informationen jeweils manuell eingeben werden müssten).
- Zum Finden von Geräten werden meist einfache Techniken eingesetzt, die aber beispielsweise leicht von Firewalls blockiert oder durch andere Konfigurationseinstellungen bei den Geräten ineffektiv gemacht werden können (es gibt jedoch auch andere, ausgefeiltere Techniken zum Finden von Geräten, wie in Kapitel *3 Grundlagen des Network Discoverys* gezeigt wird).

Die Behandlung all dieser Unzulänglichkeiten ist die Motivation zur Erstellung der vorliegenden Arbeit.

## 1.3 Stand der Forschung

Das Thema Network Discovery wurde bereits mehrfach wissenschaftlich behandelt (jedoch mit den zuvor genannten Unzulänglichkeiten). Dieser

Abschnitt listet exemplarisch einige wichtige Publikationen auf diesem Gebiet (sowie deren Fokus) auf, auf denen auch diese Arbeit aufbaut. Eine komplette Liste aller verwendeten Quellen befindet sich in Anhang *A Literaturverzeichnis*. In späteren Kapiteln wird im Kontext des jeweiligen Themas detaillierter auf die entsprechenden Quellen verwiesen.

Die meisten Arbeiten behandeln eines von drei Themen: entweder das Finden von Geräten oder die Bestimmung der Layer 3 oder der Layer 2 Topologie. Zur letzten Gruppe gehört eine Reihe von Papers, die sich mit komplexen Algorithmen zur Inferenz der Topologie mittels der Informationen aus den Address Forwarding Tables von Switches befassen (wobei diese jedoch bekannt sein müssen). Diese Werke bauen teilweise aufeinander auf und dazu gehören (in entsprechender Reihenfolge) die Arbeiten von Lowekamp et al. ([44] bzw. [45]), Breitbart et al. ([10]), Bejerano et al. ([5]) und Sun et al. ([78]). Meiss et al. und Nazir et al. verwenden in [53] bzw. [58] dazu noch weitere Mechanismen, um Switches auch automatisch zu erkennen. Stott et al. wählen in [76] hingegen einen anderen Ansatz zur Bestimmung der Layer 2 Topologie, nämlich die zusätzliche Verwendung der Spanning Tree Informationen. Zedler et al. legen in [89] besonderes Augenmerk auf den physischen Standort von Geräten. Black et al. beschreiben in [8] eine komplett andere Methode zum Finden der Layer 2 Topologie, nämlich das Senden von Probe Packets über Daemons, die auf jedem Computer laufen müssen.

Die zweite große Gruppe von Arbeiten beschäftigt sich mit der Bestimmung der Layer 3 Topologie, vor allem der des Internets. Dazu gehören die Werke von Siamwalla et al. ([72]), Govindan et al. ([23]), Huffaker et al. ([28]) und Adhikari et al. ([1], mit besonderem Fokus auf die operationale Layer 3 Topologie), welche alle auf das Senden von Probe Packets setzen, während Stott et al. in [77] Routing Tables über SNMP auslesen. Das bei der Bestimmung der Layer 3 Topologie des Internets auftretende Problem der IP-Aliasse einzelner Router sprechen Spring et al. in [74] an.

Die dritte Gruppe von Arbeiten behandelt das Auffinden von Geräten in einem Netzwerk. Eine simple Methode dazu präsentieren Schönwälder et al. in [69]. Wood et al. kombinieren in [86] bereits mehrere Ansätze, während Nazir et al. in [58] der Idee eines kompletten Systems, das verschiedene Techniken integriert, am nächsten kommen. Arkin ([2]) und Lyon ([47] und [48]) beschäftigen sich in ihren Werken mit ausgefeilten Scanning-Techniken, um Geräte unter Umgehung von Firewalls zu finden.

Andere Arbeiten, wie etwa die von Vigna et al. ([82]), handeln vom Aufbau von Discovery-Systemen aus bereits bestehenden Tools, die je eine spezielle Aufgabe erfüllen. Schließlich beschäftigt sich eine letzte Gruppe von Arbeiten jeweils mit einem ganz spezifischen Teilaspekt des Discoverys. Bellovin

et al. behandeln in [6] zum Beispiel das Finden von Geräten hinter einem NAT Router. Weihua et al. erforschen in [85] die Möglichkeiten von ICMP, beispielsweise zum OS Fingerprinting. Montigny et al. erörtern in [57] rein passives Network Discovery.

## 1.4 Zielsetzung

Die Ziele dieser Arbeit ergeben sich großteils direkt aus der Auseinandersetzung mit den in Abschnitt *1.2 Motivation* genannten Unzulänglichkeiten bestehender Ansätze. Folgende Punkte werden in dieser Arbeit behandelt bzw. umgesetzt:

- Es wird eine Einführung in die wichtigsten Bereiche der Thematik geliefert (und nicht nur abgegrenzte Aspekte behandelt).
- Bestehende Discovery-Techniken von diversen Quellen werden an einer Stelle zusammengefasst und es wird ein Überblick über sie und ein Vergleich zwischen ihnen präsentiert. Daraus ergibt sich eine Referenz und ein Leitfaden zu der Auswahl von zur Verfügung stehenden Methoden bei der Entwicklung eines Systems bzw. Teilsystems für das Network Discovery.
- Ausgewählte Techniken werden in einem konkreten Algorithmus zusammengefasst und es wird gezeigt, wie sich einzelne Aspekte ergänzen und die vorgestellten Methoden auch tatsächlich eingesetzt werden können. Ein besonderes Augenmerk liegt hier auf einer ganzheitlichen Perspektive, wobei die (u.a. in anderen Arbeiten vorgestellten) Teile zu einem Ganzen zusammengefügt werden.
- Der genannte Algorithmus wird auch in Form eines eigens im Rahmen dieser Arbeit erstellten Referenzsystems (genannt *Network Explorer*) implementiert um zu zeigen, dass die am Papier vorgestellten Techniken auch in der Praxis funktionieren. Die Praxistauglichkeit der Software wird weiter unterstrichen, indem Ergebnisse aus dem experimentellen Einsatz präsentiert und analysiert werden.
- Die erstellte Software ist ein eigenständiges System, das nicht bestehende Tools verknüpft oder auf diesen aufbaut, sondern komplett neu entwickelt wurde. Daher hat es einerseits über alle Aspekte des Discoverys die volle Kontrolle und zeigt außerdem beispielhaft, wie ein solches System von Grund auf entworfen und implementiert werden kann.
- Sowohl im vorgestellten Algorithmus, als auch in der Software, die ihn umsetzt, werden Technologien mit praktischer Relevanz (z.B. VLANs,

Link Aggregation, virtuelle Router, virtuelle Maschinen etc.), wie sie heute Teil eines praktisch jeden größeren LANs sind, berücksichtigt.

- Ein Kernpunkt des Algorithmus' und des Programms ist das Discovery bei minimalen Vorkenntnissen. Das heißt, das Discovery soll möglichst selbstständig, ohne Vorkenntnisse über das Netzwerk und mit minimalen nötigen Eingabedaten möglich sein (denn zu viele nötige Vorkenntnisse würden die Idee des automatischen Findens von Informationen ad absurdum führen).

- Große Aufmerksamkeit wird darauf gelegt, stets eine Verbindung zu „real world" Situationen und Problemstellungen herzustellen sowie die praktische Anwendung und Anwendbarkeit der präsentierten Themen immer im Auge zu behalten. Anstatt theoretischer Korrektheit oder Vollständigkeit von Discovery-Algorithmen (die nur erreicht werden können, wenn unrealistische Einschränkungen gemacht werden bzw. gewisse Sonderfälle unberücksichtigt bleiben, wie z.B. in [45] anerkannt wird), steht bei dieser Arbeit eindeutig der praktische Nutzen im Vordergrund.

## 1.5 Abgrenzung

Analog zur Zielsetzung soll auch festgehalten werden, was explizit *nicht* Thema dieser Arbeit ist. Folgende Punkte werden dezidiert ausgeschlossen bzw. nicht berücksichtigt:

- Das Thema der Arbeit und die Aufgabe des Programms ist nur das Network Discovery (Finden von Geräten und Informationen über diese) und weder das Network Management, noch das Network Monitoring oder die Erstellung von Statistiken usw.

- Diese Arbeit beschränkt sich rein auf das Finden von Geräten und deren Verbindungen in einem abgegrenzten Netzwerk (z.B. einer Firma oder einer Universität) und nicht etwa im Internet.

- Weiters werden nur Netzwerke, die Ethernet und TCP/IP (und dabei IPv4) verwenden, sowie SNMP-Kommunikation über SNMPv1 berücksichtigt, da dies die in ihrem Bereich dominierenden Technologien sind bzw. um das Programm nicht unnötig zu verkomplizieren.

- Es wird versucht, möglichst alle Technologien mit praktischer Relevanz (z.B. VLANs, Link Aggregation, virtuelle Router, virtuelle Maschinen etc.) zu berücksichtigen, wo eine detaillierte Behandlung aber den Rahmen der Arbeit sprengen würde, wird nur auf weitere Literatur verwiesen.

- Sonderfälle werden, wo es passend ist, aufgezeigt, nicht alle möglichen Sonderfälle werden aber aufgelistet, behandelt oder berücksichtigt (da dies ebenfalls den Rahmen der Arbeit sprengen würde).
- Der vorgestellte Algorithmus ist keine Entwicklung einer komplett neuen Technik, sondern fügt teilweise bereits vorher (in anderen Arbeiten) beschriebene Methoden zusammen.
- Weiters setzen der Algorithmus bzw. das Programm nur ausgewählte und nicht alle der in dieser Arbeit beschriebenen Techniken um. Die verwendeten Methoden wurden dabei so selektiert, sodass grundlegende Informationen zu Geräten und der Topologie erfasst werden können, wobei möglichst wenige Vorkenntnisse notwendig sein sollen und der Algorithmus möglichst einfach gehalten werden können soll.
- Das Programm erfasst (auf Befehl des Anwenders) sozusagen einen Schnappschuss des Netzwerkes (innerhalb der Zeitspanne eines Discovery-Prozesses). Es findet keine kontinuierliche Überwachung statt, alte Daten (von vorangegangenen Durchläufen) werden nicht aufgehoben (d.h. es gibt keine History-Funktionen oder -Auswertungen).
- Weiters werden nur beispielhafte bzw. nur die wichtigsten Daten (z.B. Adressen, Gerätetypen, Topologie etc.) und nicht alle möglichen Daten und (Konfigurations-)Details erfasst. Es soll gezeigt werden, wie konzeptuell Geräte und deren Verbindungen untereinander gefunden werden können, es sollte aber kein Tool zum Auslesen aller überhaupt irgendwie verfügbaren Informationen gebaut werden.
- Das Programm dient generell nur als „Proof of Concept“, es enthält daher keine komfortablen Features für den „produktiven“ Betrieb (es ist nicht Multi-User-fähig, hat kein Web-Interface, keine zentrale Datenbank, es läuft nur auf *Microsoft Windows* etc.).
- Diese Arbeit ist keine Auflistung, Analyse oder Vergleich von bestehender Software (dies wurde schon zu Genüge in anderen Arbeiten vorgenommen).

## 1.6 Aufbau der Arbeit

Die Arbeit ist (nach dieser Einleitung) wie folgt gegliedert:

Als Erstes liefert Kapitel *2 Einführung in Computernetzwerke* eine kurze Einführung in die zum späteren Verständnis nötigen generellen Konzepte und Funktionsweisen von Computernetzwerken, bevor das eigentliche

Thema des Network Discoverys behandelt werden kann. Nach einer Behandlung der Grundlagen (*2.1 Grundlagen*), wird die logische Struktur der Netzwerkkommunikation anhand des OSI-Modells (*2.2 Das OSI-Modell*) vorgestellt. Darauf werden die verschiedenen Typen von Geräten in einem Netzwerk, inklusive deren Funktion (*2.3 Netzwerkkomponenten*), sowie die Arten, wie diese (physisch oder logisch) miteinander verbunden sein können (*2.4 Netzwerktopologien*) behandelt. Danach wird auf die verbreitetsten Technologien auf Layer 2 (*2.5 Ethernet*) und Layer 3/4 (*2.6 TCP/IP*) eingegangen. Abschließend werden einige weitere, für das Discovery besonders relevante Protokolle betrachtet (*2.7 Weitere Protokolle*).

Danach kann in Kapitel *3 Grundlagen des Network Discoverys* die eigentliche Thematik dieser Arbeit behandelt werden. Nach der Definition von Begriffen und deren Zuordnung zueinander und zu verwandten Konzepten (*3.1 Grundlagen*), werden grundlegende mögliche Ansätze beim Network Discovery vorgestellt (*3.2 Ansätze*). Danach werden konkrete Discovery-Techniken, gruppiert nach dem Protokoll auf dem sie basieren, in den Abschnitten *3.3* bis *3.9* präsentiert. Darauf werden noch einige weitere mögliche Techniken bzw. Datenquellen im Discovery genannt, ohne jedoch zu sehr ins Detail zu gehen (*3.10 Weitere Techniken*). Weiters werden einige wichtige Sonderfälle, die beim Discovery berücksichtigt werden müssen, mit Hinweisen zu deren Behandlung, aufgelistet (*3.11 Sonderfälle*). Die vorgestellten Methoden werden noch einmal zusammenfassend betrachtet, verglichen und somit ein Leitfaden geliefert, wofür sich welche eignet (*3.12 Leitfaden*).

Kapitel *4 Der Discovery-Algorithmus* wählt einige der zuvor vorgestellten Discovery-Techniken aus und fügt diese in einem Algorithmus zusammen und zeigt so, wie ein komplettes Discovery-System entwickelt werden kann. Nach der Beschreibung allgemeiner Aspekte dieses Algorithmus' (*4.1 Grundlagen*), getroffener Annahmen, nötiger Voraussetzungen und vorhandener Einschränkungen (*4.2 Voraussetzungen*) sowie der Ein- und Ausgabedaten, welcher der Algorithmus benötigt bzw. liefert (*4.3 Ein- und Ausgabedaten*), wird zuerst eine Übersicht über den groben Ablauf des Algorithmus' (*4.4 Übersicht*) und dann eine detaillierte Beschreibung seiner einzelnen Schritte (*4.5 Infrastructure Discovery*, *4.6 Host Discovery* und *4.7 Topology Discovery*) präsentiert. Abschließend werden die Eckpunkte des Algorithmus' sowie dessen Verhalten in bestimmten Fällen zusammenfassend betrachtet (*4.8 Zusammenfassung*).

Darauf folgend wird im Kapitel *5 Umsetzung des Algorithmus'* die eigens im Rahmen dieser Arbeit erstellte Software, welche den zuvor beschriebenen Algorithmus umsetzt, vorgestellt. Nach einer Betrachtung deren allgemeiner Eigenschaften (*5.1 Übersicht*), werden deren Funktionen (*5.2 Funktio-*

*nen*), Architektur (*5.3 Architektur*) sowie das verwendete Datenmodell (*5.4 Datenmodell*) präsentiert.

Anschließend wird in Kapitel *6 Experimentelle Ergebnisse* beschrieben, wie sich das System in der Praxis bewährt hat. Es werden die Umgebung, in der getestet wurde (*6.1 Versuchsumgebung*), die getesteten Szenarien (*6.2 Versuchsaufbau*) sowie die dabei gelieferten Resultate dargestellt und analysiert (*6.3 Ergebnisse*). Einige beim praktischen Einsatz gemachte besondere Beobachtungen werden ebenfalls nicht vorenthalten (*6.4 Besondere Beobachtungen*). Danach wird auf Basis der zuvor vorgestellten Ergebnisse das System als Ganzes und dessen Praxistauglichkeit bewertet (*6.5 Bewertung*).

Kapitel *7 Einsatzszenarien* bietet einerseits Anregungen, wie das entwickelte System in der vorliegenden Version konkret in der Praxis eingesetzt werden könnte (*7.1 Derzeitige Anwendungen*) sowie beschreibt andererseits, welche Erweiterungen möglich wären und mit welchen Anwendungen diese verbunden wären (*7.2 Erweiterungen*).

Abschließend wird in Kapitel *8 Zusammenfassung* ein Resümee über die in dieser Arbeit erzielten Ergebnisse gezogen.

Anhang *A Literaturverzeichnis* enthält eine Liste der bei der Erstellung dieser Arbeit verwendeten Quellen.

Anhang *B Network Explorer Handbuch* bietet einen Leitfaden für Benutzer des entwickelten Programmes. Beschrieben werden dessen allgemeine Eigenschaften (*B.1 Beschreibung*), Voraussetzungen für den Einsatz (*B.2 Voraussetzungen*), Schritte zur Installation (*B.3 Installation*) sowie die Bedienung von dessen Funktionen (*B.4 Bedienung*).

# Kapitel 2

# Einführung in Computernetzwerke

Dieses Kapitel bietet eine Einführung in die Konzepte, Begriffe und Funktionsweisen im Bereich der Computernetzwerke. Diese Einführung beinhaltet jedoch nur die zum weiteren Verständnis nötigen Grundlagen und ist nicht als komplette Darstellung der Thematik anzusehen (genaue Paket- und Datenformate, Algorithmen im Detail und dergleichen werden hier nicht behandelt, es sei denn, es ergeben sich aus diesen Punkten Implikationen im weiteren Verlauf der Arbeit).

Sofern nicht anders angeführt, sind die in diesem Kapitel dargestellten Sachverhalte [79] entnommen. Auf dieser Einführung aufbauend, kann im folgenden Kapitel *3 Grundlagen des Network Discoverys* auf die eigentliche Thematik der Arbeit – das Auffinden von Geräten in einem Netzwerk sowie von Informationen über diese Geräte und über die Struktur von deren Verbindungen untereinander – eingegangen werden.

## 2.1 Grundlagen

Ein *Computernetzwerk* (kurz auch nur *Netzwerk* oder *Netz* bzw. auf Englisch *Network* genannt) ist ein Zusammenschluss von eigenständigen Computersystemen, die miteinander kommunizieren und Daten austauschen können.

Damit ein Computer über das Netzwerk mit anderen kommunizieren kann, wird einerseits eine physische Hardware-Schnittstelle (*Interface*) zum Netzwerk benötigt sowie andererseits die dazu passende Software, welche die Kommunikation steuert. Damit verschiedene Computer untereinander Daten austauschen können, müssen sie sich an gewisse vordefinierte Regeln (genannt *Protokolle*) halten, damit sie einander auch „verstehen“ können. Hierauf wird im nächsten Abschnitt *2.2 Das OSI-Modell* näher eingegangen.

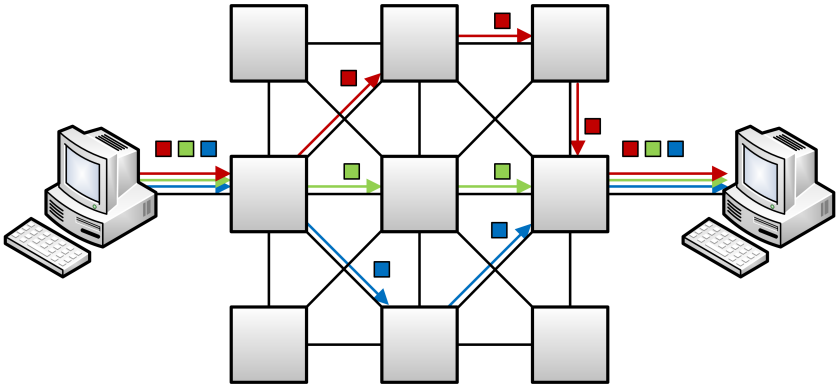

**Abbildung 2.1:** Packet Switching

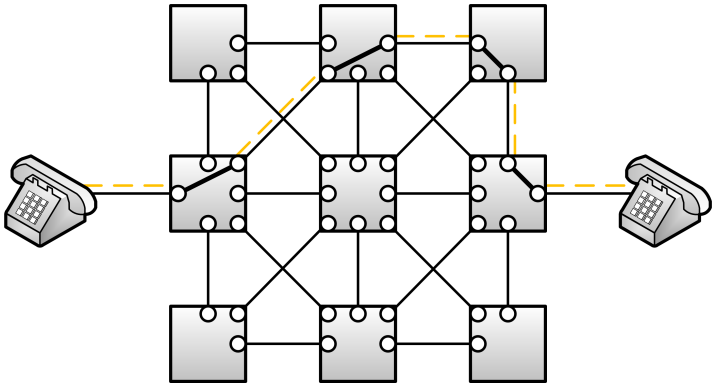

**Abbildung 2.2:** Circuit Switching

Heutzutage wird bei Computernetzen meist sogenanntes *Packet Switching* eingesetzt (dargestellt in *Abbildung 2.1*). Dies bedeutet, dass die zu übertragenden Daten vom Sender in *Pakete* (*Packets*) aufgeteilt werden. Diese Pakte werden dann einzeln versandt und von einem Knoten im Netzwerk zum nächsten weitergeschickt – wobei nicht alle Pakete zwangsläufig den gleichen Weg nehmen müssen –, bis sie den Empfänger erreichen, der die Pakte wieder zur ursprünglichen Nachricht zusammensetzt. Der Vorteil dieser Vorgehensweise liegt darin, dass es meist mehrere Wege für die Daten zwischen zwei Computern in einem Netzwerk gibt, sodass selbst beim Ausfall von einem oder mehreren Knoten im Netzwerk, die Daten über einen anderen Weg ihr Ziel erreichen können, wodurch eine gegenüber Fehlern tolerante und robuste Kommunikation möglich ist.

Im Gegensatz dazu steht das *Circuit Switching* (siehe *Abbildung 2.2*), wie es früher z.B. bei der Telefonie eingesetzt wurde, wobei vor der eigentlichen Kommunikation zuerst eine Verbindung zwischen den Teilnehmern hergestellt und ein Übertragungskanal zwischen ihnen durchgeschaltet wird, über

den dann kommuniziert wird. Hier ist der Nachteil, dass die Kommunikation unmöglich wird, sobald dieser Kanal unterbrochen wird.

Es gibt viele Möglichkeiten Netzwerke zu klassifizieren, wie etwa nach dem Übertragungsmedium (z.B. Kupferkabel, Glasfaserkabel, Funk, Mikrowellen, Infrarot), eingesetzter Technologie bzw. Protokollen (z.B. Ethernet, Token Ring, FDDI, UMTS, TCP/IP, IPX/SPX), funktionaler Architektur (z.B. Client-Server, Peer-to-Peer), Topologie (z.B. Bus, Ring, Baum, Stern) usw. Eine weit verbreitete Art, Netzwerke zu klassifizieren, ist nach ihrer Ausdehnung. Hierbei oft verwendete Klassen sind (neben anderen) sogenannte *Wide Area Networks* (*WAN*) und *Local Area Networks* (*LAN*).

LANs sind von ihrer Ausdehnung her beschränkt, z.B. auf ein Gebäude, den Standort einer Firma, den Campus einer Universität usw. – daher auch das Wort „lokal“ in der Bezeichnung. Eine klassische Anwendung von LANs ist, in einer Büroumgebung Workstations, Server und Drucker zu verbinden.

WANs hingegen dehnen sich über einen größeren Bereich aus – wie das Wort „weit“ in der Bezeichnung nahe legt – und ermöglichen so die Kommunikation über Ländergrenzen oder gar Kontinente hinweg. Einzelne, jeweils eigenständige LANs können über ein WAN miteinander verbunden werden und so auch über weite Distanzen Daten austauschen. Das bekannteste Beispiel für ein WAN ist das *Internet*.

Diese Arbeit beschränkt sich auf das Auffinden von Geräten in *LANs* (z.B. dem Netzwerk einer Firma oder einer Universität), da diese meist zu einer einzelnen Organisation gehören und das Netzwerk einer einheitlichen Verwaltung untersteht. Weiters werden nur *Ethernet*- und *TCP/IP*-Netzwerke betrachtet (d.h. Netzwerke in denen diese Technologien eingesetzt werden), da dies die in diesem Bereich dominanten Technologien sind. In den folgenden Abschnitten wird auf diese näher eingegangen.

## 2.2 Das OSI-Modell

Wie bereits erwähnt, müssen sich Computer in einem Netzwerk an Regeln bezüglich des Datenaustausches halten, damit eine Kommunikation erfolgen kann. Diese Regeln werden *Protokolle* genannt. Es gibt eine Unmenge verschiedenster Protokolle, welche die unterschiedlichsten Aufgaben erledigen. Meist sind an einer einzelnen Datenübertragung eine Vielzahl von Protokollen beteiligt, die sich die Arbeit untereinander aufteilen. Zusammengehörige Protokolle bilden *Protokollfamilien* (*Protocol Suites*).

Die *ISO* (*International Organization for Standardization*) hat ein Sieben-Schichten-Modell entwickelt, das so genannte *OSI-Modell* (*Open Systems*

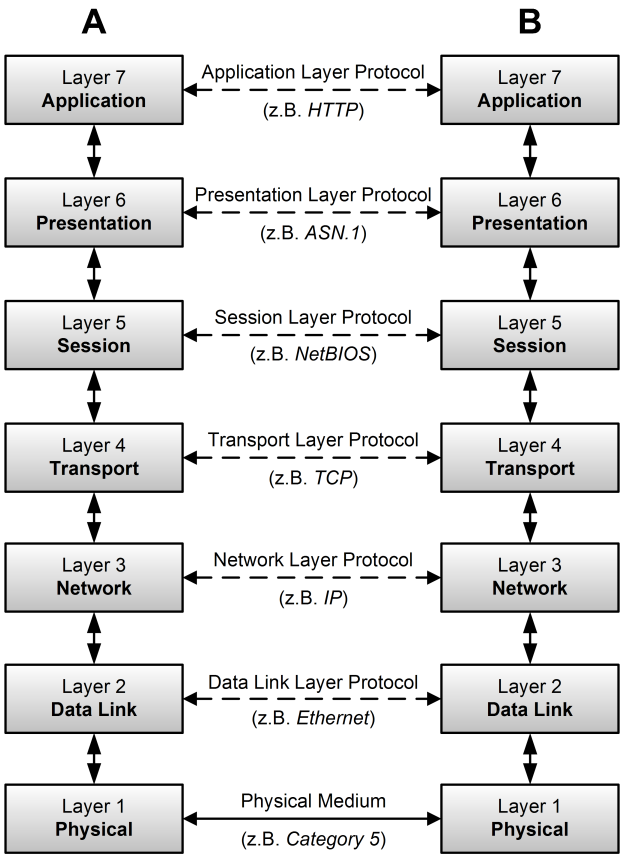


**Abbildung 2.3:** Das OSI-Modell

*Interconnection*, definiert in [34]), welches die Kommunikationsfunktionen innerhalb eines Netzwerkes bzw. von Netzwerken untereinander aufgliedert, definiert und somit die Kommunikation strukturiert. Jedes zum Datenaustausch notwendige Protokoll wird einer der sieben Schichten (*Layers*) zugeordnet, wobei jede Schicht nur existiert, um der darüberliegenden Dienste anzubieten. Dieses Modell ist in *Abbildung 2.3* dargestellt.

Durch die Aufteilung der gesamten mit der Kommunikation zusammenhängenden Funktionalität auf diese Schichten wird einerseits die Komplexität beim Entwurf der Kommunikation verringert und andererseits können einzelne Unterfunktionen geändert (bzw. Protokolle gegen andere ausgetauscht) werden, ohne die gesamte Funktionsweise zu stören (vorausgesetzt die Schnittstellen zwischen den Ebenen bleiben gleich).

Ein Programm, das eine Nachricht übertragen möchte, übergibt diese der obersten Schicht. Die Nachricht wird dann in jeder Schicht verarbeitet und zum jeweils nächsten Layer weitergereicht, bis sie auf der untersten Ebene schließlich über das Netzwerk gesendet wird. Die unterste Schicht stellt das

eigentliche Übertragungsmedium (z.B. ein Glasfaserkabel) dar, über welches die Daten (z.B. im Form von Lichtimpulsen) verschickt werden. In der Empfängerstation wird die Nachricht auf der untersten Schicht entgegengenommen und bis zum empfangenden Programm durch die Schichten hinaufgereicht.

Während sich jede Schicht zwar der darunterliegenden bedient, um die Daten weiterzureichen, so erfolgt konzeptuell die Kommunikation immer zwischen den Schichten auf der selben Ebene. Das heißt, Schicht $n$ auf Computer $A$ kommuniziert logisch gesehen mit Schicht $n$ auf Computer $B$ (wobei das Ergebnis der Kommunikation der Schicht $n+1$ zur Verfügung gestellt wird und die Schicht $n-1$ dazu benutzt wird, um die Kommunikation durchzuführen). Dabei müssen sich die Schichten auf Ebene $n$ auf beiden Computern an die Regeln für diese Schicht halten, man bezeichnet diese daher auch *Layer n Protokoll*.

Werden Daten von einer höheren Schicht zu einer niedrigeren gereicht, werden sie in dieser meist in kleinere Datenpakete zerteilt (es wird generell, wie im vorigen Abschnitt beschrieben, Packet Switching eingesetzt) und diese erhalten zusätzliche, für diese bestimmte Schicht notwendige, Informationen angehängt (*Header* und *Trailer*). Die eigentlichen Daten werden sozusagen von jeder Schicht immer weiter „eingepackt“. Beim Hinaufreichen werden diese Informationen wieder entfernt und einzelne Datenfragmente wieder zusammengesetzt. Die softwaremäßige Implemetierung der so übereinander „gestapelten“ Protokollschichten nennt man *Protocol Stack*. Die von den Protokollen zu Steuerungszwecken zusätzlich erzeugten Daten (im Vergleich zu den eigentlichen Nutzdaten) werden *Protocol Overhead* genannt.

*Abbildung 2.4* stellt das „Verpacken“ von Daten durch die involvierten Protokolle am Beispiel der Übertragung einer Webseite (welche den *HTTP Body* ausmacht) grafisch dar. In diesem Beispiel wird der TCP/IP-Stack (siehe [79]) verwendet, welcher die Schichten 5 und 6 nicht benutzt und die Ebenen 1 und 2 als eine Schicht ansieht, daher sind diese auch nicht eingezeichnet.

Es gibt zwar eine eigene Protokollfamilie, die streng nach dem OSI-Modell entwickelt wurde, diese hat sich jedoch auf Grund der Dominanz vom älteren TCP/IP im Internet nicht durchgesetzt (siehe ebenfalls [79]). Dennoch ist die Funktionalität des OSI-Modells in fast allen Protokollfamilien zu finden, wobei aber oft zwei oder drei OSI-Schichten übersprungen oder zu einer zusammengefasst werden. Trotzdem liefert das OSI-Modell ein sehr gutes *Referenzmodell* zum Analysieren und Vergleichen von Protokollfamilien. Weiters dient es dazu, um einzelne Konzepte im Bezug auf die Kommunikation als Ganzes sowie im Bezug zu anderen Konzepten ein- und zuordnen zu können. Darin liegt auch die Relevanz des Modells für diese Arbeit.

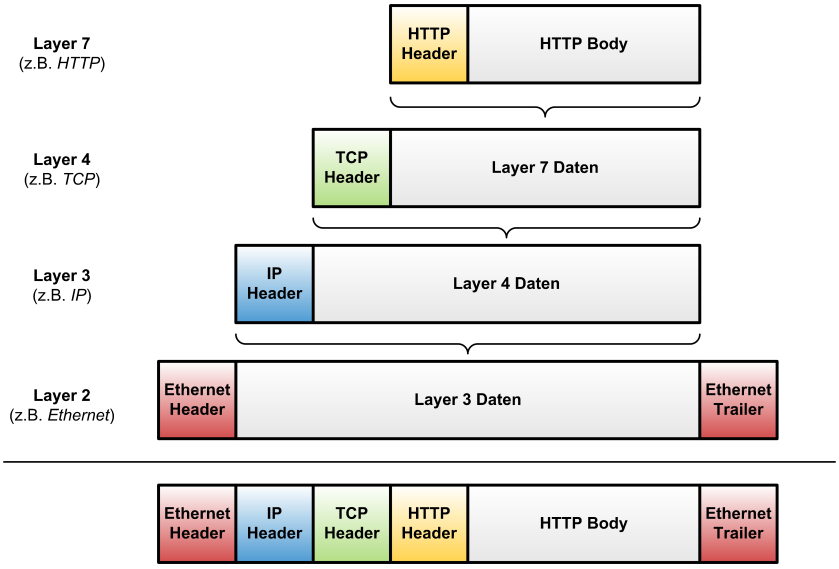


**Abbildung 2.4:** Protocol Overhead

Im Folgenden werden die einzelnen Schichten des OSI-Modells näher betrachtet.

### 2.2.1 Layer 7: Application Layer

Die oberste Schicht ist der *Application Layer* (*Anwendungsschicht*). Diese Ebene bildet die Schnittstelle zwischen den auf einem Computer laufenden Programmen und dem Netzwerk. Auf dieser Schicht agierende Protokolle bieten Dienste wie Hypertext-Übertragung, Dateitransfer, E-Mail, Network Management etc. und definieren die dafür notwendigen Datenformate. Alle unter dieser Schicht liegenden Layer existieren nur, um diese Aktivitäten zu ermöglichen.

Beispiele für Protokolle dieses Layers sind z.B. das *Hypertext Transfer Protocol* (*HTTP*), *File Transfer Protocol* (*FTP*), *Simple Mail Transfer Protocol* (*SMTP*), *Simple Network Management Protocol* (*SNMP*) und viele mehr (siehe [79]).

### 2.2.2 Layer 6: Presentation Layer

Der *Presentation Layer* (*Darstellungsschicht*) stellt sicher, dass die Daten vom Application Layer richtig interpretiert werden können. Es ist zum Beispiel möglich, dass zwei in einem Netzwerk liegende Computer unterschiedliche Methoden zur Kodierung des Zeichensatzes verwenden, wie etwa *American Standard Code for Information Interchange* (*ASCII*, siehe *RFC 20* [13]) und *Extended Binary Coded Decimals Interchange Code* (*EBCDIC*,

siehe [80]). Deshalb wird auf dem Presentation Layer beider Computer eine gemeinsame Repräsentationsform ausgehandelt. Der Presentation Layer bringt dann alle vom Application Layer erhaltenen Daten in diese Form und vice-versa. Ver- und Entschlüsselung finden auch auf dieser Schicht statt.

Die *Abstract Syntax Notation One* (*ASN.1*, siehe [35]) – welche z.B. von SNMP zur Kodierung der Daten verwendet wird – ist ein weiteres Beispiel für ein Protokoll dieser Ebene.

### 2.2.3 Layer 5: Session Layer

Der *Session Layer* (*Kommunikationssteuerungsschicht*) koordiniert die Kommunikation zwischen den Teilnehmern und verwaltet den Dialog. Es wird z.B. ausgehandelt, ob Daten nur in eine Richtung (*simplex*), abwechselnd in beide Richtungen (*half duplex)* oder gleichzeitig in beide Richtungen (*full duplex*) gesendet werden können. Passwortabfragen oder der Wiederaufbau von Sitzungen, falls in einer der darunterliegenden Schichten ein Fehler aufgetreten ist, fallen auch in den Aufgabenbereich dieser Schicht.

Das Protokoll *NetBIOS* (siehe *RFC 1001* [59]) arbeitet z.B. auf diesem Layer. In den meisten Protokollfamilien wird diese Schicht jedoch kaum verwendet bzw. die von ihr angebotenen Dienste sind im Layer 4 eingearbeitet.

### 2.2.4 Layer 4: Transport Layer

Der *Transport Layer* (*Transportschicht*) stellt sicher, dass die ihm vom Session Layer übergebenen Daten bis zum Empfänger übertragen werden. Der Transport Layer erhält von der darüberliegenden Schicht eine *Nachricht* (*Message*), teilt diese in kleinere *Pakete* (*Packets*) auf und gibt diese an den Layer 3 zur Übertragung weiter.

Manche auf dieser Ebene operierende Protokolle stellen auch eine fehlerfreie und korrekte Übertragung sicher: Falls Pakete bei der Übertragung verloren gehen oder beschädigt werden, wird dies in der Empfängerstation erkannt und es wird veranlasst, dass diese Pakete noch einmal von der Senderstation geschickt werden. Auch der Fall, dass Pakete nicht in der richtigen Reihenfolge empfangen werden, kann von Protokollen auf diesem Layer behandelt werden.

Das *Transmission Control Protocol* (*TCP*, siehe *RFC 793* [65]) und das *User Datagram Protocol* (*UDP*, siehe *RFC 768* [62]) agieren zum Beispiel auf dieser Ebene.

### 2.2.5 Layer 3: Network Layer

Der *Network Layer* (*Vermittlungsschicht*) ist dafür zuständig, dass die vom Transport Layer erhaltenen Pakete ihr Ziel erreichen, d.h. den richtigen Weg durch das Netz nehmen. Um Absender und Empfänger von Paketen identifizieren zu können, erhalten Geräte auf diesem Layer logische *Netzwerkadressen* (z.B. *IP-Adressen*).

Der Network Layer garantiert nur, dass die Pakete richtig *geroutet* werden, d.h. den Weg durch ein (oder mehrere) Netzwerk(e) zum Ziel finden. Dass auch alle Pakete korrekt und vollständig ankommen, ist, wie ausgeführt, Aufgabe des Transport Layers.

Das prominenteste Beispiel für ein Protokoll dieser Ebene ist das *Internet Protocol (IP), siehe RFC 791 [64].* Da IP (auf Layer 3) meist zusammen mit TCP (auf Layer 4) eingesetzt wird, spricht man von der *TCP/IP* Protokollfamilie (siehe Abschnitt *2.6 TCP/IP* für nähere Details).

### 2.2.6 Layer 2: Data Link Layer

Der *Data Link Layer* (*Sicherungsschicht*) bestimmt, in welcher Form die vom Network Layer erhaltenen Pakete über das Übertragungsmedium (z.B. Kupferkabel, Glasfaserkabel, Funk etc.) übertragen werden. Die Pakete werden in sogenannte *Frames* weiter aufgeteilt und über das Medium gesendet. Wie diese Frames auszusehen haben, wird vom verwendeten Netzwerksystem bestimmt (z.B. *Ethernet*, siehe [79]).

Der Data Link Layer überwacht das Aktivieren und Deaktivieren des Sendevorgangs, bestimmt die Zugriffsmethode (z.B. *CSMA/CD*, siehe ebenfalls [79]), erkennt und korrigiert Übertragungsfehler von Frames usw. Um Absender und Empfänger von Frames identifizieren zu können, besitzen Interfaces auf dieser Ebene *Hardwareadressen* (z.B. *MAC-Adressen*). Ein Gerät hat somit (ein oder mehrere) Hardware- und Netzwerkadressen, die voneinander unabhängig sind (und auf verschiedenen Ebenen des OSI-Modells agieren).

### 2.2.7 Layer 1: Physical Layer

Der *Physical Layer* (*Bitübertragungsschicht*) bestimmt die physikalischen Voraussetzungen für die Datenübertragung, wie z.B. Kabeltypen, Steckertypen, Widerstände usw. Vereinfacht ausgedrückt, stellen die verwendeten Kabel den Physical Layer dar.

Die vom Data Link Layer erhaltenen Frames werden als reiner *Bitstrom* über das Übertragungsmedium gesandt. Multiplexing-Techniken wie *Frequency*

*Division Multiplexing* (*FDM*) oder *Time Division Multiplexing* (*TDM*) werden ebenfalls auf dieser Schicht durchgeführt.

## 2.3 Netzwerkkomponenten

Nachdem im vorigen Abschnitt die Struktur der Kommunikation in einem Netzwerk beleuchtet wurde, werden in diesem die einzelnen Typen von Komponenten (Arten von Geräten) in einem Netz betrachtet und diese in Bezug zur Kommunikationsstruktur gesetzt. Es wird besonderes Augenmerk auf solche Komponenten bzw. diejenigen Gesichtspunkte einzelner Komponenten gelegt, die eine Relevanz im Network Discovery haben.

### 2.3.1 Interfaces

*Interfaces* – auch genannt *Netzwerkadapter*, *Netzwerkkarten*, *Network Interface Controllers* (*NIC*) etc. – sind, wie am Anfang dieses Kapitels bereits erwähnt, die physische Schnittstelle zwischen einem Computer und dem Netzwerk. Sie ermöglichen dem Computer also, mit dem Netzwerk verbunden zu sein und darüber zu kommunizieren. Konzeptionell befinden sich Interfaces auf *Layer 1* und *Layer 2* des OSI-Modells.

Geräte können auch mehr als nur ein Interface besitzen (sie sind dann also mehrfach mit einem oder mehreren Netzwerken verbunden) und werden in diesem Fall *multi-homed* genannt. Es gibt mehrere Gründe, warum ein Gerät mehrere Interfaces haben kann bzw. haben soll.

Geräte, die mehrere Netzwerke bzw. Teile eines Netzwerkes miteinander verbinden (wie Router und Switches, siehe dazu die folgenden Abschnitte), müssen natürlich auch mehrere Interfaces haben. Mehrere Interfaces können aber auch zur Ausfallssicherheit (z.B. bei Servern) eingesetzt werden, damit die Netzwerkverbindung aufrecht bleibt, auch wenn ein Interface ausfällt. Dabei können die Interfaces auch so konfiguriert werden, dass sie sich so verhalten, als ob das Gerät nur ein Interface hätte – in Wirklichkeit wird abwechselnd das eine und das andere verwendet (und die Verbindung ist gegen den Ausfall eines Interfaces sicher). Ähnlich dazu können auch mehrere physische Interfaces zu einem virtuellen zusammengefasst werden (und verhalten sich dann analog wie ein einzelnes), um den Datendurchsatz zu erhöhen. Das heißt, es werden mehrere Interfaces parallel genutzt, um mehr Daten gleichzeitig versenden zu können. Dies ist ebenfalls bei Servern oder zwischen Switches gebräuchlich und wird *Link Aggregation* (oder auch *Trunking*, *Bonding*, *Port Channel* etc.) genannt (siehe [32]).

Weiters können Geräte auch rein *virtuelle Interfaces* haben (Daten werden auf einem physischen Interface empfangen und intern an das virtuelle

Interface weitergeleitet). Diese entstehen nicht nur, wie oben angeführt, aus dem Zusammenschluss von physischen Interfaces, sondern können auch unabhängig von physischen Interfaces existieren, z.B. um dahinter bestimmte Funktionen anzubieten. Es gibt z.B. Geräte, die gleichzeitig Switch und Router sind. Dabei hat der Switch physische Interfaces für alle angeschlossenen Geräte sowie ein virtuelles Interface, hinter dem sich die Routing-Funktionen verbergen – ganz so, als ob an diesem Interface ein separater, physischer Router angeschlossen wäre, nur dass dieser Router nur virtuell in der Software ein und desselben Geräts existiert (siehe [17]).

Normalerweise ist jedem (physischen und virtuellen) Interface eine eindeutige Hardwareadresse zugeordnet. Diese Adressen werden verwendet, um den Absender und Empfänger von Frames anzugeben, die jeweils von einem Interface zu einem anderen geschickt werden. Siehe Abschnitt *2.5 Ethernet* für nähere Details dazu.

Aus Sicht des Network Discoverys können in einem ersten Schritt nur einzelne Interfaces erkannt werden (wobei auch nicht zwischen physischen und virtuellen Interfaces unterschieden werden kann), da im Netz nur die angesprochenen Hardwareadressen auftauchen (Details folgen im Kapitel *3 Grundlagen des Network Discoverys*). Daher ist auch kaum feststellbar, ob es sich z.B. bei zwei gefundenen Interfaces um zwei separate Geräte oder um ein Gerät mit zwei Interfaces handelt (und ob beide physisch oder etwa eines nur virtuell ist). Umgekehrt kann ein Gerät auch mehrere physische Interfaces als ein virtuelles Interface betreiben (d.h. die physischen Interfaces verwenden alle die gleiche Hardwareadresse). In diesem Fall wird nur ein einzelnes Interface erkannt.

Erst mit zusätzlichen Informationen (die von den Geräten selbst abgefragt werden müssen – falls diese diese Information anbieten – oder durch den Einsatz spezieller Heuristiken) können Interfaces auch zusammengefasst und einzelnen Geräten zugeordnet werden. Mehr Details dazu folgen in Abschnitt *3.3.3 Interface Tables*.

### 2.3.2 Übertragungsmedien

Die *Übertragungsmedien* (z.B. verschiedenste Typen von Kupferkabeln und Glasfaserkabeln mit unterschiedlichsten Eigenschaften oder auch einfach nur Luft oder Vakuum, wie im Falle einer Funkübertragung) verbinden die Geräte (genauer gesagt die Interfaces der Geräte) in einem Netzwerk und dienen, wie der Name schon sagt, zur Übertragung der Daten und befinden sich auf *Layer 1* des OSI-Modells.

Im Discovery können verwendete Übertragungsmedien nur indirekt erkannt werden (z.B. falls ein Gerät die Informationen über seine Interfaces

preisgibt und diese Informationen auch Hinweise auf das angeschlossene Übertragungsmedium enthalten), sind dafür aber auch nicht relevant (außer diese Information wird explizit gewünscht). Aus Sicht des Discoverys ist es nur wichtig, dass Geräte verbunden sind (bzw. wie sie von der Struktur her untereinander zusammenhängen, siehe dazu Abschnitt *2.4 Netzwerktopologien*), nicht aber worüber.

### 2.3.3 Hubs

Unter einem *Hub* kann man sich in etwa eine Verteilerdose (analog zu einem Stromverteiler) vorstellen. Er hat mehrere Ein-/Ausgänge (*Ports*), an denen Stationen sternförmig angeschlossen werden können. Kommt an einem Port ein Signal an, wird dieses bei allen anderen Ports hinausgesandt. So können an einem Hub angeschlossene Stationen miteinander kommunizieren.

Hubs arbeiten auf dem *Layer 1* des OSI-Modells. Da sie *passiv* sind (eingehende Signale zwar duplizieren, aber nicht selbst als eigenständiges Gerät ansprechbar sind), können sie im Discovery nur indirekt erkannt werden, z.B. indem mehrere Geräte an einer Stelle gefunden werden, wo nur ein Gerät sein dürfte, woraus sich schließen lässt, dass dazwischen ein Hub sein muss (vgl. [76]).

Die Erkennung von Hubs im Discovery ist einerseits wichtig, da dies Auswirkungen auf die Topologie hat (siehe Abschnitt *2.4 Netzwerktopologien*), aber andererseits auch nicht einfach, da es Fälle geben kann, in denen ein Hub vorliegen könnte oder auch nicht (da Hubs nicht direkt erkennbar oder ansprechbar sind) bzw. irreführende Informationen gegeben sind (mehr Details und Techniken dazu folgen im Abschnitt *4.7 Topology Discovery*).

### 2.3.4 Switches

*Switches* sind (ähnlich wie Hubs) Verteiler (analog zu einem Telefonverteiler) mit mehreren Interfaces, welche in diesem Zusammenhang auch *Ports* genannt werden. Wird ein Frame an einem Port empfangen, wird bestimmt, wer der Empfänger dieses Frames ist. Der Switch weiß im Regelfall (siehe nächster Absatz), an welchem Port welcher Empfänger hängt und sendet das Frame dann nur über den entsprechenden Port weiter. Durch Switches gelangen Frames also nur zu dem vorbestimmten Empfänger und werden (im Gegensatz zu Hubs) nicht unnötig an andere unbeteiligte Stationen ausgesandt.

Switches sind selbstlernend und erkennen anhand der Hardwareadressen der Absender der empfangenen Frames, welche Station an welchem Port hängt und merken sich dies. Sollte ein Switch (noch) nicht wissen, an welchem

Port ein gesuchter Empfänger angeschlossen ist, sendet er das Frame (wie ein Hub) über alle Ports.

Switches arbeiten auf dem *Layer 2* des OSI-Modells. Sie sind *aktiv*, d.h. sie führen selbst Berechnungen durch (welche Daten über welchen Port gesandt werden sollen) und sind meist als eigenständige Geräte ansprechbar (so sie die entsprechenden Funktionen unterstützen). Früher hießen Switches mit nur zwei Ports *Bridges* (da sie, wie eine Brücke, zwei Geräte oder zwei Teile eines Netzwerkes verbinden). Daher taucht auch heute noch dieser Begriff im Zusammenhang mit Switches auf (auch wenn moderne Switches viel mehr als nur zwei Ports haben).

Switches spielen beim Discovery eine entscheidende Rolle bei der Bestimmung der Topologie, da sie, wie zuvor erwähnt, die an ihnen angeschlossenen Geräte kennen. Dies wird im Detail in den Abschnitten *3.3.6 Address Forwarding Tables* und *4.7 Topology Discovery* behandelt.

Meist sind Switches nicht direkt an die Endgeräte angeschlossen, sondern deren Ports sind zuerst über ein sogenanntes *Patch Panel* mit je einer Netzwerkdose (die sich z.B. in einem Büro befindet) verbunden – man sagt auch, der Switch Port ist zu der Dose *gepatcht* –, an welcher letztendlich das Gerät hängt. Da die Netzwerkdosen aber passiv sind, ist es aus Sicht des Discoverys so, als ob die Geräte direkt mit den Switches verbunden wären. Soll bekannt sein, welches Gerät sich an welcher Dose (und somit in welchem Raum) befindet (siehe auch Kapitel *7 Einsatzszenarien*), so muss die Information (die sogenannte *Patchung*), welcher Switch Port mit welcher Dose verbunden ist, separat erfasst werden. Dies kann jedoch im Allgemeinen nur händisch geschehen, da, wie gerade angeführt, eine Netzwerkdose nicht direkt erkennbar ist (außer es werden spezielle Patch Panels verwendet, worauf hier jedoch nicht näher eingegangen werden soll).

Mehrere Switches werden oft zu einem *Verteiler* zusammengefasst. Dies ist meist ein Schrank mit mehreren Einschüben (*Slots*), von denen jeder eine Switch-Einheit ist. Dies erlaubt Verteiler je nach Bedarf zu erweitern, indem neue Einschübe hinzugefügt werden. Manche Verteiler (je nach Hersteller und Typ) präsentieren sich im Discovery als ein Gerät (ein Switch), während bei manch anderen jeder Slot als eigener Switch ansprechbar ist. Dies führt dazu, dass beim Discovery in dem einem Fall ein Switch mit vielen Ports und im anderen Fall viele Switches mit weniger Ports gefunden werden.

Es gibt zwei Möglichkeiten, wie die Ports eines Verteilers bzw. Switches intern miteinander verdrahtet sein können. Bei *Matrix Switches* ist jeder Port mit jedem anderen verbunden. Alternativ dazu können die Ports an einem gemeinsamen Bus (der sogenannten *Backplane*) angeschlossen sein

und so über diesen Daten zueinander schicken. Im zweiten Fall hat jeder Switch, zusätzlich zu den Ports an denen die Endgeräte hängen, weitere *Backplane Ports*, die jeden Slot mit jedem anderen verbinden. Dies sei hier erwähnt, da dies in speziellen Fällen Auswirkungen auf das Discovery haben kann, wie in Abschnitt *6.4 Besondere Beobachtungen* beschrieben.

### 2.3.5 Router

*Router* erfüllen konzeptionell ähnliche Aufgaben wie Switches, jedoch auf der nächsthöheren Ebene des OSI-Modells, d.h. auf *Layer 3*. Ein Router arbeitet nicht mit Frames und Hardwareadressen, sondern mit Paketen und Netzwerkadressen.

Jedes Paket (das u.U. aus mehreren Frames bestehen kann) hat u.a. eine Zieladresse (eine Netzwerkadresse). Der Router entscheidet mittels interner Tabellen (*Routing Tables*) und eigener Routing-Protokolle, welche Station der nächste Schritt (*Hop*) auf dem Weg zum endgültigen Ziel ist. Die Frames des Pakets erhalten dann als Ziel die Hardwareadresse dieses nächsten Schritts und werden über den entsprechenden Port ausgesandt. Sind die Frames beim nächsten Router angekommen, wird wieder entschieden, wie der weitere Weg des Pakets auszusehen hat. Dies wiederholt sich so lange, bis das Ziel erreicht wurde, d.h. der nächste Schritt kein weiterer Router, sondern das eigentliche Zielgerät ist (vgl. *Abbildung 2.1*).

Wie bereits erwähnt, wird auf Layer 3 mit logischen Netzwerkadressen gearbeitet, die beliebig vergeben werden können, wodurch sich ergibt, dass die logische (Netzwerkadressen) und physische (Hardwareadressen) Netzstruktur (Topologie) voneinander unabhängig sind. Einzelne Netzwerkadressen werden Netzen bzw. Subnetzen (siehe Abschnitt *2.6 TCP/IP*) zugeordnet bzw. in diesen zusammengefasst, Router vermitteln zwischen diesen Netzen.

Für das Discovery sind Router unter zwei Gesichtspunkten wichtig: Einerseits kennen sie die angeschlossenen Netze. Diese Information wird zur Bestimmung der Layer 3 Topologie benötigt. Andererseits arbeiten Router sowohl mit Netzwerk- als auch mit Hardwareadressen und können diese einander zuordnen. Im Discovery werden, je nachdem welche Methode gerade zum Sammeln von Daten eingesetzt wird und auf welcher Ebene diese Methode arbeitet, entweder Netzwerk- oder Hardwareadressen gefunden. Diese müssen danach einander zugeordnet werden und es muss bestimmt werden, was zu welchen Interfaces und welchen Geräten gehört, wobei u.a. die Informationen von Routern behilflich sein können (konkrete Techniken dazu werden in Kapitel *3 Grundlagen des Network Discoverys* behandelt).

Router müssen nicht zwangsläufig eigene physische Geräte sein (da ihre Aufgabe im Prinzip nur ist, die Ziel-Hardwareadresse von empfangenen Frames

gegen jene des nächsten Schritts auszutauschen), sondern können auch rein virtuell (als Layer 3 Funktion) auf einem physischen Switch existieren. Je nach Hersteller und Typ eines derartigen Switches mit Routing-Funktion, kann dieser sich nach außen hin als ein Gerät (mit zwei Funktionen) oder als zwei Geräte (wobei eines davon nur virtuell existiert) darstellen. Mehr dazu folgt im Abschnitt *3.3 Simple Network Management Protocol (SNMP)*.

### 2.3.6 Endgeräte

Als *Endgeräte* werden in dieser Arbeit alle anderen Typen von Geräten bezeichnet, die nicht zur Infrastruktur des Netzwerks (wie Switches und Router) gehören, sondern eine „eigentliche“ Funktion (auf *Layer 7*) für den Endanwender erfüllen, wie z.B. *Workstations*, *Laptops*, *Server*, *Drucker*, *IP-Telefone*, *Videokonferenzsysteme* und diverse Arten von Spezialgeräten (*Network Probes*, *Unterbrechungsfreie Stromversorgungen*, *Gebäudeautomatisationssysteme*, *Zeiterfassungssysteme* etc.).

Die Aufgabe des Network Discoverys ist es, nicht nur Informationen über die Netzwerk-Infrastruktur zu finden, sondern auch möglichst alle vorhandenen Endgeräte zu erkennen. Dabei ist zu berücksichtigen, dass es verschiedenste Arten von Geräten von den unterschiedlichsten Herstellern gibt. Daher muss ein Discovery-System auch mit einem derart heterogenen Umfeld umgehen können.

Wie bereits in Abschnitt *2.3.1 Interfaces* beschrieben, werden Geräte nicht direkt im Netzwerk gefunden, sondern nur deren Interfaces (die einzelnen Geräten zugeordnet werden müssen). Wie auch bereits ausgeführt, sind Geräte über (ein oder mehrere) Hardware- und Netzwerkadressen identifiziert, die ebenfalls entsprechend zugeordnet werden müssen.

Vor allem im Server-Bereich gibt es einige Sonderfälle, die beim Discovery zu berücksichtigen sind. Im Abschnitt Abschnitt *2.3.1 Interfaces* wurden bereits *multi-homed* Geräte, *virtuelle Interfaces* und *Link Aggregation* behandelt.

Ein anderer Spezialfall sind sogenannte *Cluster*. Bei diesen verhalten sich mehrere physische Computer so, als ob sie ein einziges Gerät wären (wobei es verschiedene Techniken gibt, dies genau zu implementieren, was jedoch nicht Thema dieser Arbeit ist). Verschiedene Einsatzszenarien bzw. Cluster-Typen sind beispielsweise *High Performance Cluster* zur Erhöhung der Rechenleistung (da mehrere Maschinen für Berechnungen zur Verfügung stehen), *Load Balancing Cluster* zur Lastverteilung (auf mehrere Geräte) sowie *High Availability Cluster* zur Erhöhung der Ausfallssicherheit (wenn ein Computer ausfällt, können andere dessen Aufgabe übernehmen). Aus Sicht des Discoverys können sich, je nach eingesetzter Clustering-Technik, Cluster als

ein einziges Gerät darstellen oder es können die einzelnen Maschinen des Clusters separat gefunden werden. Dies muss entsprechend beachtet werden.

Ein weiterer Sonderfall sind sogenannte *virtuelle Maschinen* (*VM*). Hierbei handelt es sich, wie der Name schon sagt, um Geräte (genannt *Guests*), die nur virtuell auf einer physischen Maschine (genannt *Host*) existieren, wobei meist mehrere Guests auf einem Host laufen. Aus Sicht des Discoverys ist es schwierig zu erkennen, welche Maschinen physisch und welche nur virtuell sind (und welche Guests zu welchen Hosts gehören), da normalerweise jede virtuelle Maschine eine eigene Hardwareadresse verwendet. Frames mit den verschiedenen Hardwareadressen der Guests werden zwar alle über das selbe physische Interface des Hosts verschickt, da beim Discovery aber, wie mehrfach erwähnt, Geräte bzw. Interfaces anhand von Hardwareadressen identifiziert werden, sieht es zunächst so aus, als ob es sich um mehrere Maschinen handeln würde (konzeptionell sind es auch mehrere, wovon aber nur die Hosts auch physisch existieren). Genauso verleitet es z.B. zur Annahme, wenn mehrere Geräte an einem Switch Port gefunden werden, dass zwischen diesen und dem Port ein Hub liegt. In Wirklichkeit kann es aber sein, dass daran nur eine physische Maschine (und kein Hub) angeschlossen ist und die anderen gefunden Geräte nur virtuell auf dem Host existieren.

### 2.3.7 Weitere Komponenten

In diesem Abschnitt werden weitere Komponenten betrachtet, die sich nicht direkt in eine der zuvor behandelten Kategorien einordnen lassen, aber dennoch eine Relevanz im Discovery haben.

*Firewalls* dienen dem Schutz von Netzwerken gegen unerlaubten Zugriff. Sie arbeiten meist auf Layer 3 und filtern ein- und ausgehende Pakete nach bestimmten Kriterien, wie z.B. Absender, Empfänger oder Protokoll. Falls zwischen dem Computer, von dem aus nach anderen Geräten gesucht wird (der Discovery-Maschine), und den zu findenden Geräten eine Firewall liegt, kann es sein, dass diese die Pakete von der Discovery-Maschine filtert (d.h. verwirft und nicht an deren Ziel weiterleitet) und dadurch das Discovery behindert. Für die Discovery-Maschine sieht es so aus, als ob die gesuchten Zielgeräte nicht existieren würden (da von ihnen keine Antwort kommt), während in Wirklichkeit nur die Firewall die Kommunikation blockiert. Sollte es nicht möglich sein, die Firewall passend konfigurieren zu lassen, ist das Einzige, was in so einem Fall gemacht werden kann, zu versuchen, die Firewall mit diversen Tricks (siehe Kapitel *3 Grundlagen des Network Discoverys*) zu umgehen, d.h. derartige Pakete zu senden, die von der Firewall nicht gefiltert werden.

*WLAN Access Points* erlauben es Computern sich drahtlos über Funk mit dem Netzwerk zu verbinden. Die so angeschlossenen Geräte befinden sich in einem *Wireless LAN* (*WLAN*). Die Access Points sind Sende- und Empfangsstationen und bilden die Schnittstelle zwischen dem WLAN und dem LAN. Sie sind Sonderformen von Switches, die auf der einen Seite normal mit dem LAN verkabelt sind und auf der anderen Seite ein Funk-Interface besitzen.

*WAN Router* sind Sonderformen von Routern, die ein LAN mit einem WAN verbinden. Während normale Router meist auf allen Seiten das gleiche Netzwerksystem verwenden (z.B. *Ethernet*), verbinden WAN Router unterschiedliche Netzwerksysteme (z.B. *Ethernet* auf der LAN-Seite und *ATM*, *ISDN* oder *Frame Relay* auf der WAN-Seite), da für die Datenübertragung über weite Strecken andere Technologien eingesetzt werden. WAN Router bilden also die Schnittstelle eines LANs nach außen.

## 2.4 Netzwerktopologien

Neben den verschiedenen Typen von Komponenten in einem Netzwerk sowie der Struktur der Kommunikation unter diesen, ist ein weiterer Aspekt, wie diese Komponenten miteinander verbunden sind – dies wird *Topologie* genannt. Darunter versteht man das Muster, nach dem Computer (physisch oder logisch) zusammengeschlossen sind.

*Abbildung 2.5* zeigt exemplarisch einige Topologien (von links oben: *Bus*, *Stern*, *Ring*, *vollvermascht*, *Baum*).

Bei der *Bus-Topologie* sind alle Stationen an ein gemeinsames Kabel angeschlossen. Jede Station kann direkt mit jeder anderen kommunizieren, da sich das Signal von einer Station im Kabel in beide Richtungen ausbreitet. Vorteile dieser Topologie sind, dass die Struktur sehr modular ist, das Hinzufügen oder Abklemmen von Geräten während des Betriebs möglich ist, der Ausfall einer Station den Betrieb nicht stört und das Senden direkt von der Quelle zum Ziel möglich ist. Nachteile sind jedoch, dass einerseits das komplette Netz bei Beschädigung des Kabels ausfällt und andererseits immer nur eine Station zu einer Zeit senden kann. Sollten mehrere Stationen gleichzeitig senden, kommt es zu Signalkollisionen und die Datenübertragung ist gestört. Daher müssen spezielle Übertragungstechniken (z.B. *CSMA/CD*, siehe [79]) angewandt werden, um das zu erkennen und zu verhindern. Auf Grund dieser Gegebenheiten ist auch die maximale Länge des Kabels begrenzt.

In einer *Stern-Topologie* sind alle Stationen an einer zentralen Verteilereinheit (Hub oder Switch) angeschlossen über welche die gesamte Kommunikation läuft – die einzelnen Stationen können nicht unmittelbar mit-

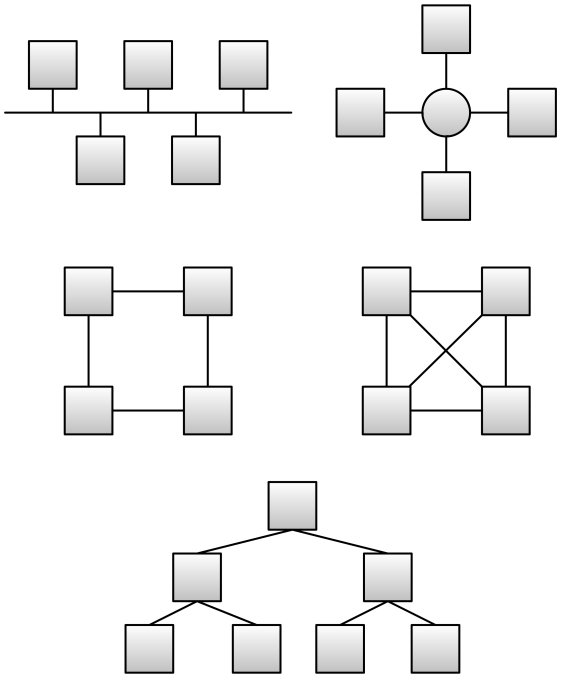

**Abbildung 2.5:** Netzwerktopologien

einander kommunizieren. Diese Topologie erfordert zusätzliche Hardware, wird dafür aber von Kabelbrüchen oder Ausfällen einzelner Stationen nicht beeinträchtigt (dafür aber umso mehr vom Ausfall der Verteilereinheit). Ist das Netzwerk *vollgeswitcht* (d.h. es gibt keine Hubs, sondern nur Switches) und unterstützen alle Kabel (sowie die Stationen) gleichzeitige bidirektionale Kommunikation, kann es auch zu keinen Signalkollisionen kommen.

Bei der *Ring-Topologie* sind alle Knoten, wie der Name nahe legt, ringförmig miteinander verbunden, wobei die Kommunikation immer nur in eine Richtung erfolgt. Jede Station besitzt einen eindeutigen Vorgänger und Nachfolger. Eine Nachricht wird immer vom Vorgänger empfangen und zum Nachfolger weitergeleitet. Dadurch kann es keine Signalkollisionen geben und es ist auch nicht möglich das Netzwerk zu überlasten, falls zu viele Stationen gleichzeitig senden wollen. Jedoch ist bei Ausfall einer Station oder eines Kabels das ganze Netzwerk lahmgelegt.

Ein Netzwerk wird als *vollvermascht* bezeichnet, wenn jede Station mit jeder anderen verbunden ist. Dadurch kann jede Station direkt mit jeder anderen kommunizieren, ohne dass es zu Signalkollisionen kommen kann und ohne dass der Ausfall einer Station oder eines Kabels das Netzwerk stört. Dies ist jedoch dafür mit extrem hohem Aufwand bei der Verkabelung verbunden. Sind nur manche Stationen mit manch anderen verbunden, so spricht man analog dazu von einem *teilvermaschten* Netzwerk.

Bei der *Baum-Topologie* sind alle Stationen hierarchisch angeordnet, jede Station ist mit der in der Hierarchie höher stehenden verbunden und verzweigt zu den hierarchisch niedrigeren. Daraus ergibt sich eine Baumstruktur. Hier kann es ebenfalls zu keinen Signalkollisionen kommen, jedoch bedingt der Ausfall eines Kabels oder einer Station, dass der unter dieser Station liegende Teilbaum unerreichbar wird.

Die sogenannte *Strukturierte Verkabelung* (u.a. standardisiert nach der Europäischen Norm *EN 50173*) dient als Vorgabe bei der Errichtung von LANs und ist eine spezielle Form der Stern-Topologie, die zusätzlich eine hierarchische Gliederung hat. So sind z.B. alle Computer eines Stockwerks mit einem Switch verbunden, die Switches aller Stockwerke sind wiederum z.B. über eine leistungsfähige Glasfaser-Steigleitung zusammengeschlossen. Die einzelnen Gebäude eines Areals sind wieder miteinander verbunden etc. So bilden kleine Einheiten zusammen größere, die mit ihresgleichen wieder noch größere Einheiten bilden. Mit dieser Vorgangsweise ist es möglich, Gelände hierarchisch und – wie der Name sagt – strukturiert zu verkabeln.

In der Praxis existiert keine dieser Topologien in ihrer Reinform. Meist wird ein Netzwerk grob nach den Regeln der Strukturierten Verkabelung aufgebaut, wodurch sich zwischen den Switches eine Baumstruktur ergibt. Die Endgeräte sind sternförmig an den einzelnen Switches angeschlossen. Zusätzlich gibt es oft sogenannte *Backup Links* zwischen Switches (Verbindungen, die normalerweise inaktiv sind, aber aktiviert werden, falls die normale Verbindung ausfällt), wodurch sich ein vermaschter Charakter ergibt. Andere Topologien (z.B. Bus oder Ring) sind nur mehr historisch oder in Spezialfällen von Bedeutung.

Aus Sicht des Discoverys ist es dessen Aufgabe – neben dem Finden von Geräten und von Informationen zu diesen – zu bestimmen, wie die Geräte untereinander verbunden sind, d.h. wie die Topologie konkret aussieht. Eine Topologie gibt es dabei nicht nur auf der Ebene von physischen Verbindungen (Layer 1 bzw. 2 im OSI-Modell), sondern auch nach logischen Verbindungen (Layer 3), z.B. die Zuordnung von Geräten zu Subnetzen und die Router zwischen diesen (mehr dazu in Abschnitt *2.6 TCP/IP*).

Da die Topologie nicht direkt vom Netzwerk oder den einzelnen Geräten abgefragt werden kann, müssen verschiedene Informationen von diversen Quellen gesammelt und zu einem konsistenten Bild zusammengefügt werden. Dazu wurden in der Literatur einige Algorithmen vorgeschlagen, z.B. in [78], [76], [5], [53], [45] und [10]. Dieses Thema wird im Abschnitt *4.7 Topology Discovery* näher behandelt.

## 2.5 Ethernet

Dieser Abschnitt betrachtet die grundlegenden Aspekte und Funktionsweisen von *Ethernet*, dem heutzutage dominanten Netzwerksystem für LANs, wobei wiederum besonderes Augenmerk auf die für das Discovery relevanten Punkte gelegt wird.

### 2.5.1 Grundlagen

*Ethernet* bezeichnet eine Reihe von verwandten Netzwerksystemen, die alle auf *Layer 2* des OSI-Modells operieren, die im Laufe der Zeit weiterentwickelt und verbessert wurden und heute der de facto Standard für LANs sind. Hier soll nicht auf die Unterschiede und Feinheiten der verschiedenen Varianten eingegangen, sondern die gemeinsamen Grundlagen vorgestellt werden (siehe [32] für den kompletten Standard).

Bei Ethernet ist jedes Interface im Netzwerk eindeutig durch eine 48 Bit (6 Byte) lange Hardwareadresse, genannt *MAC-Adresse* (*Media Access Control*), identifiziert, welche generell als Abfolge von sechs Zahlen (eine pro Byte) im Hexadezimalformat (wodurch ein Byte mit zwei Zeichen dargestellt werden kann), getrennt durch einen Bindestrich oder einen Doppelpunkt, dargestellt wird (z.B. `08:00:20:AE:FD:7E`). Diese MAC-Adresse jedes Interfaces ist gewöhnlicherweise bereits vom Hersteller voreingestellt (kann aber meist auch per Software geändert werden bzw. können Interfaces auch Frames mit anderen MAC-Adressen als Absender verschicken, wie es z.B. VM Hosts machen, wenn sie Frames der Guests versenden).

Damit MAC-Adressen immer eindeutig sind, auch wenn die Netzwerkkarten von verschiedenen Herstellern stammen, besteht jede MAC-Adresse konzeptuell aus zwei Teilen: Die ersten 24 Bits (3 Bytes) sind die sogenannte Herstellerkennung (auch genannt *Organizationally Unique Identifier* oder *Vendor ID*), wobei jeder Firma, die Interfaces herstellt, eine eigene Kennung für ihre Produkte zugewiesen ist. Die restlichen 24 Bits (3 Bytes) sind eine eindeutige Kennung jedes Interfaces des jeweiligen Herstellers.

Die spezielle MAC-Adresse `FF:FF:FF:FF:FF:FF` bezeichnet kein bestimmtes Interface, sondern ist die sogenannte *Broadcast Address*. An diese Adresse geschickte Frames werden an alle Interfaces weitergeleitet und von allen Interfaces verarbeitet. Manchmal taucht (bei den Einstellungen von Interfaces, nicht in versandten Frames) die MAC-Adresse `00:00:00:00:00:00` auf. Diese steht für eine ungültige bzw. nicht definierte MAC-Adresse.

Normalerweise verarbeiten Interfaces nur Frames, die als Empfänger die eigene MAC-Adresse (oder die Broadcast-Adresse) eingetragen haben. Viele Interfaces lassen sich jedoch auch im sogenannten *Promiscuous Mode* betrei-

ben, in dem sie alle empfangenen Frames – egal ob diese an sie gerichtet sind oder nicht – verarbeiten. Dies kann im Discovery nützlich sein, wenn die auf einem Netzwerk gesendeten Daten (*Traffic)* abgehört werden sollen.

Dennoch kann auch im Promiscuous Mode ein Interface nicht den kompletten Traffic eines Netzwerkes empfangen, sondern nur denjenigen des *Segments* an dem das Interface angeschlossen ist. Segmente werden auch *Collision Domains* genannt, da in den ersten Versionen von Ethernet die Geräte in einer Bus-Topologie alle an dem selben Kabel angeschlossen waren, wodurch es zu Signalkollisionen kommen konnte, wenn mehrere Stationen gleichzeitig senden wollten (siehe [79]). Daher musste ein spezielles Verfahren, genannt *Carrier Sense Multiple Access with Collision Detection* (*CSMA/CD*) – auf das hier jedoch nicht näher eingegangen werden soll – verwendet werden, um derartige Kollisionen zu vermeiden bzw. damit umzugehen.

Alle Geräte, die sich im selben Segment bzw. der selben Collision Domain befinden, können direkt (auf Layer 1, also ohne Switches dazwischen) miteinander kommunizieren (da sie am selben Kabel hängen), aber es kann auch zu Kollisionen von deren Signalen kommen – daher auch der Name Collision Domain. Befindet sich ein Switch zwischen den Geräten, so teilt dieser das Netz in mehrere Segmente, da er Frames zuerst auf einem Port empfängt, verarbeitet und dann auf einem anderen Port wieder ausgibt. Ein Hub tut dies nicht, da er empfangene Signale einfach dupliziert und auf allen Ports ausgibt, wodurch es weiterhin zu Kollisionen kommen kann und alle angeschlossenen Geräte sich weiterhin im selben Segment befinden.

Daher kann, wie erwähnt, selbst im Promiscuous Mode maximal nur der Traffic auf dem lokalen Segment mitgehört werden (Traffic von anderen Segmenten, der nicht für das lokale Segment bestimmt ist, wird von den Switches gar nicht erst in das lokale Segment weitergeleitet). Ist ein Netzwerk vollgeswitcht (was heutzutage meist der Fall ist), so schrumpfen die Segmente auf jeweils das Kabel zwischen jedem Gerät und dem Switch Port, an dem es hängt (womit die Nützlichkeit des Promiscuous Modes verloren geht). Das heißt, in jedem Segment gibt es nur mehr zwei Maschinen, das Endgerät und den Switch.

Ein weiterer Begriff in diesem Zusammenhang ist die sogenannte *Broadcast Domain*. Darunter versteht man denjenigen Bereich, an den Frames an die Broadcast-Adresse weitergeleitet werden. Broadcast-Frames werden von Switches weitergeschickt (über alle Ports), nicht jedoch von Routern. Die Broadcast Domain ist also der Bereich (oft das ganze LAN), in dem Geräte direkt (auf Layer 2, also mit Switches dazwischen) miteinander kommunizieren können. Um Geräte in anderen Broadcast Domains zu erreichen, muss

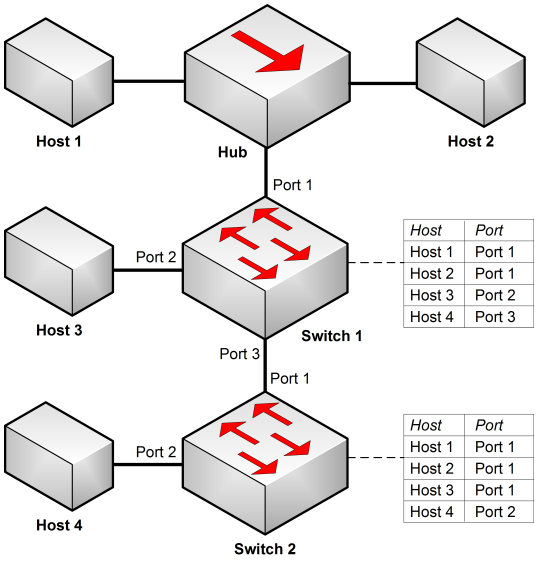


**Abbildung 2.6:** Address Forwarding Tables

die Kommunikation auf einer höheren Ebene (Layer 3) stattfinden und über einen Router gehen (wie dies geschehen soll, ist jedoch nicht mehr Aufgabe von Ethernet).

### 2.5.2 Switching

Switches in einem Ethernet Netzwerk arbeiten so, wie in Abschnitt *2.3.4 Switches* beschrieben (siehe auch [30]). Deren Funktionsweise wird auch als *Transparent Bridging* bezeichnet, da sie für die beteiligten Geräte unmerklich stattfindet. Ein Gerät schickt ein Frame an die MAC-Adresse des Empfängers und die Switches sorgen dafür, dass das Frame ins Segment des Empfängers weitergeleitet wird, wo dieser es direkt empfangen kann.

Die Information, welche Geräte – direkt oder indirekt (d.h. hinter kaskadierten Switches) – an welchem Port eines Switches angeschlossen sind, wird in den sogenannten *Address Forwarding Tables* (*AFT*), auch *Forwarding* oder *Filtering Databases* (*FDB*) genannt, gespeichert. In einem AFT gibt es einen Eintrag für jede MAC-Adresse, die der Switch von empfangenen Frames kennengelernt hat, wobei dazu gespeichert ist, an welchem Port diese Frames eingegangen sind. *Abbildung 2.6* stellt zwei Switches mit ihren AFTs dar (in der Grafik sind die Namen der Geräte in den AFTs dargestellt, in Wirklichkeit würden an dieser Stelle die MAC-Adressen der entsprechenden Interfaces der Geräte stehen).

Ports von Switches, die nicht direkt mit Endgeräten, sondern mit anderen Switches verbunden sind, werden *Uplink Ports* genannt. An einem Uplink

Port werden alle Geräte gehört, die an dem angeschlossenen Switch (oder an anderen an diesem Switch angeschlossenen Switches) hängen. Theoretisch sollte das AFT eines Switches einen Eintrag für jedes Gerät im LAN (bzw. genauer in der selben Broadcast Domain) haben, da alle Geräte entweder direkt an dem Switch oder über einen Uplink Port mit diesem verbunden sind. Praktisch sind die Informationen in den AFTs jedoch oft lückenhaft, denn damit die Informationen überall vollständig wären, müsste jeder Switch erst einmal von jedem Gerät ein Frame empfangen haben. Das heißt, die an einem Switch angeschlossenen Geräte müssten mit allen anderen Geräten kommuniziert haben. Weiters sind die Einträge in den AFTs mit einem Timeout versehen (standardmäßig 5 Minuten). Wird innerhalb dieser Zeit kein Frame mit der entsprechenden Absenderadresse empfangen, wird nach Ablauf der Zeitspanne der Eintrag entfernt. Dies dient dazu, die AFTs von Einträgen für Geräte, die abgeschaltet oder entfernt wurden, zu reinigen.

Switches können auch *Link Aggregation* (siehe [32]), wie in Abschnitt *2.3.1 Interfaces* beschrieben, zu anderen Switches oder Endgeräten (meist Servern), einsetzen. In diesem Fall enthält der Switch einen virtuellen Port für die *Link Aggregation Group*. AFT Einträge verweisen entsprechend auf diesen und nicht auf die an der Link Aggregation Group beteiligten physischen Ports.

Da Broadcast-Frames von den Switches über all ihre Ports weitergeleitet werden, darf es in der Topologie des Ethernet-Netzwerkes keine Schleifen geben, weil die Broadcast-Frames sonst endlos im Kreis geschickt werden würden (genannt *Broadcast Storm*). Daher muss eine Baum-Topologie vorliegen. Um dies zu erreichen bzw. sicherzustellen, verwenden manche Switches das *Spanning Tree Protocol* (*STP*), welches in [30] definiert wird. Dabei wird ein Switch als Wurzel (*Root*) des Baumes bestimmt und dann all jene Verbindungen zwischen den Switches deaktiviert, die zu Schleifen führen würde, sodass jeder Switch nur eine einzige aktive Verbindung besitzt, die in Richtung des Roots führt.

Für eine Erläuterung, wie das Spanning Tree Protocol genau funktioniert, um eine Baum-Topologie sicherzustellen, sei auf [30] verwiesen. Für den weiteren Verlauf ist es lediglich wichtig zu wissen, dass zu jedem Port eines Switches folgende Informationen gespeichert werden:

- Der Root Switch (*Designated Root*), welcher über den Port erreicht wird.
- Der direkt benachbarte Switch (*Designated Bridge*), der mit diesem Port verbunden ist. Direkt benachbart bedeutet, dass kein anderer Switch dazwischen ist. Hubs können jedoch durchaus dazwischen liegen, d.h. die Switches können auch über einen Hub verbunden sein.

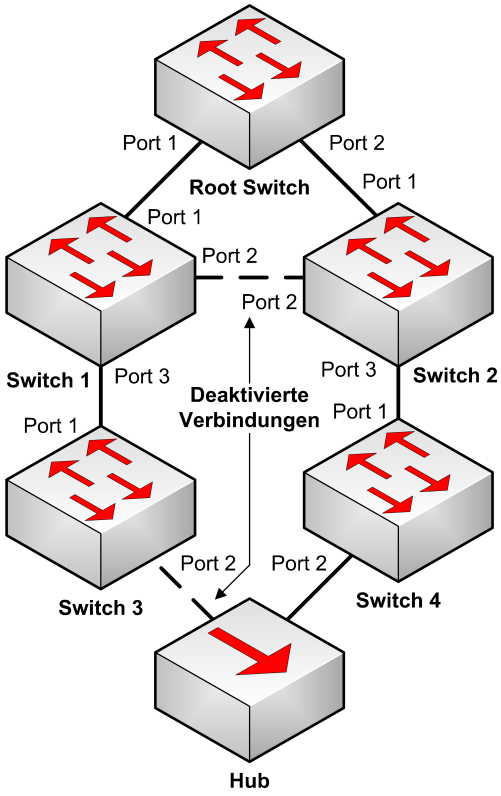


**Abbildung 2.7:** Spanning Tree Protocol

- Der Port des direkt benachbarten Switches (*Designated Port*), der mit diesem Port verbunden ist.

*Abbildung 2.7* zeigt die Funktionsweise des Spanning Tree Protocols (Verbindungen, die zu Schleifen führen würden, werden deaktiviert). *Tabelle 2.1* stellt die dazugehörigen Port Tables jedes Switches dar. In der Abbildung und in der Tabelle sind der Übersicht wegen die Namen der Switches und Ports dargestellt. In Wirklichkeit werden Switches durch eine 8 Byte Adresse (*Bridge ID*) und Ports durch eine 2 Byte Adresse (*Port ID*) identifiziert, die an den entsprechenden Stellen in der Tabelle stehen würden.

Es ist anzumerken, dass die Einträge in den Tabellen immer nur in eine Richtung verweisen (zum Root). Das heißt, dass z.B. `Port 1` von `Switch 4` auf `Port 3` von `Switch 2` verweist, aber nicht umgekehrt. Weiters gilt es anzumerken, dass zwar `Port 2` von `Switch 3` deaktiviert ist (da es sonst eine Schleife gäbe), nicht jedoch `Port 2` von `Switch 4` (da sonst die am Hub angeschlossenen Geräte nicht mehr erreichbar wären).

| Port | Des. Root | Des. Switch | Des. Port | Aktiv |
|---|---|---|---|---|
| *Switch 1* | | | | |
| Port 1 | Root Switch | Root Switch | Port 1 | ja |
| Port 2 | Root Switch | Switch 2 | Port 2 | nein |
| Port 3 | | | | ja |
| *Switch 2* | | | | |
| Port 1 | Root Switch | Root Switch | Port 2 | ja |
| Port 2 | | | | ja |
| Port 3 | | | | ja |
| *Switch 3* | | | | |
| Port 1 | Root Switch | Switch 1 | Port 3 | ja |
| Port 2 | Root Switch | Switch 4 | Port 2 | nein |
| *Switch 4* | | | | |
| Port 1 | Root Switch | Switch 2 | Port 3 | ja |
| Port 2 | | | | ja |

**Tabelle 2.1:** STP Port Table

### 2.5.3 Virtuelle LANs

Ein *Virtual Local Area Network* (*VLAN*) ist ein separates, virtuelles LAN innerhalb eines physischen LANs. VLANs werden in [31] beschrieben. Die Idee dahinter ist, Geräte nach logischen Kriterien (z.B. separate LANs für die Buchhaltung, Forschungsabteilung etc.) in VLANs zu gruppieren, unabhängig von ihrem tatsächlichen Standort, wobei sie alle am selben physischen LAN angeschlossen sind. Aus Sicht der Geräte in den verschiedenen VLANs ist es so, als ob sie mit separaten LANs verbunden wären (jedoch ohne dass wirklich separate physische LANs verlegt und installiert werden müssten). Jedes VLAN wird durch eine Nummer (*VLAN ID*) identifiziert.

VLANs bilden eigene Broadcast Domains, daher ist die Kommunikation zwischen VLANs nur über einen Router möglich. Daher ist es auch nicht möglich, ein Subnetz (siehe Abschnitt *2.6 TCP/IP*) in mehreren VLANs gleichzeitig zu betreiben (in einem VLAN kann es jedoch ohne Probleme mehrere Subnetze geben), da Geräte per Definition direkt miteinander kommunizieren (und nicht über einen Router), wenn sie im selben Subnetz sind, genau diese direkte Kommunikation aber nicht möglich ist, wenn sie in verschiedenen VLANs liegen.

Eine Möglichkeit VLANs aufzubauen ist es, jeden Switch Port einem bestimmten VLAN zuzuteilen. Alle an einem Port angeschlossenen Geräte befinden sich dann automatisch in dem entsprechenden VLAN, das dem Port zugeordnet ist. Wird ein Frame von einem Gerät an einem Switch

Port empfangen, so fügt der Switch dem Frame die Kennung (*Tag*) des entsprechenden VLANs hinzu. Damit ist dieses Frame nun gekennzeichnet, zu welchem VLAN es gehört. Dieses Frame wird nun nach den normalen Forwarding-Regeln zwischen den Switches weiter übertragen. Bevor das Frame über den letzten Port in der Kette zum Empfänger gesandt wird, entfernt der dazugehörige Switch das VLAN Tag wieder. Dadurch ist einerseits sichergestellt, dass es, während die Frames ihren Weg durch das LAN nehmen, immer klar ist, zu welchem VLAN jedes Frame gehört und andererseits, dass die Endgeräte jedoch nichts von der Präsenz der VLANs merken (da die Tags ohne deren Wissen hinzugefügt und wieder entfernt werden).

Ports, die Switches miteinander verbinden (Uplink Ports), müssen Frames aus allen VLANs übertragen. Daher werden zwischen diesen, wie im vorigen Absatz beschrieben, Frames mit einem VLAN Tag (*getaggte* Frames) verschickt, damit erhalten bleibt, zu welchem VLAN ein jedes Frame gehört. Diese Ports, die Frames aus mehreren VLANs übertragen, werden *Trunk Ports* genannt.

Es ist auch möglich, dass Geräte selbst Frames verschicken, die bereits mit einem VLAN Tag versehen sind. Dazu ein Beispiel: An einem Switch Port, der `VLAN 1` zugeordnet ist, hängt ein Hub, an dem wiederum ein PC und ein IP-Telefon angeschlossen sind (weil etwa keine separate Netzwerkdose nur für das Telefon mehr frei ist). Nun möchte man, dass alle PCs in `VLAN 1` liegen, aber alle IP-Telefone in `VLAN 2`. In der momentanen Konfiguration befinden sich aber beide in `VLAN 1`, da deren Frames automatisch dieses VLAN zugewiesen bekommen, sobald sie am Switch Port empfangen werden. Nun ist es möglich, dass das IP-Telefon Frames verschickt, die bereits mit dem `VLAN 2` getaggt sind. Werden diese Frames vom Switch Port empfangen, bleibt das schon bestehende Tag erhalten und das IP-Telefon kann somit ins `VLAN 2` senden. Damit auch umgekehrt das Empfangen funktioniert, darf der Switch Port das VLAN Tag von Frames an das IP-Telefon *nicht* entfernen (im Gegensatz zu Frames aus dem `VLAN 1`), bevor er es über den Port ausgibt. Gleichzeitig muss das IP-Telefon auch getaggte Frames empfangen können (wovon aber auszugehen ist, wenn es sie senden kann).

Damit das genannte Szenario funktioniert, müssen also Geräte einerseits VLAN getaggte Frames senden und empfangen können und andererseits müssen die Switch Ports, an denen die Geräte hängen, auch entsprechend konfiguriert sein (d.h. sie müssen einerseits eingehende, bereits getaggte Frames aus gewissen VLANs auch akzeptieren und andererseits Frames aus diesen VLANs auch *mit* Tag wieder ausgeben). Dass die Switches für solche Fälle speziell konfiguriert sein müssen, ist insofern wichtig, da sonst Geräte von sich aus den VLAN-Mechanismus umgehen könnten, indem sie getaggte Frames senden, was den Sinn von VLANs (die Trennung des Daten-

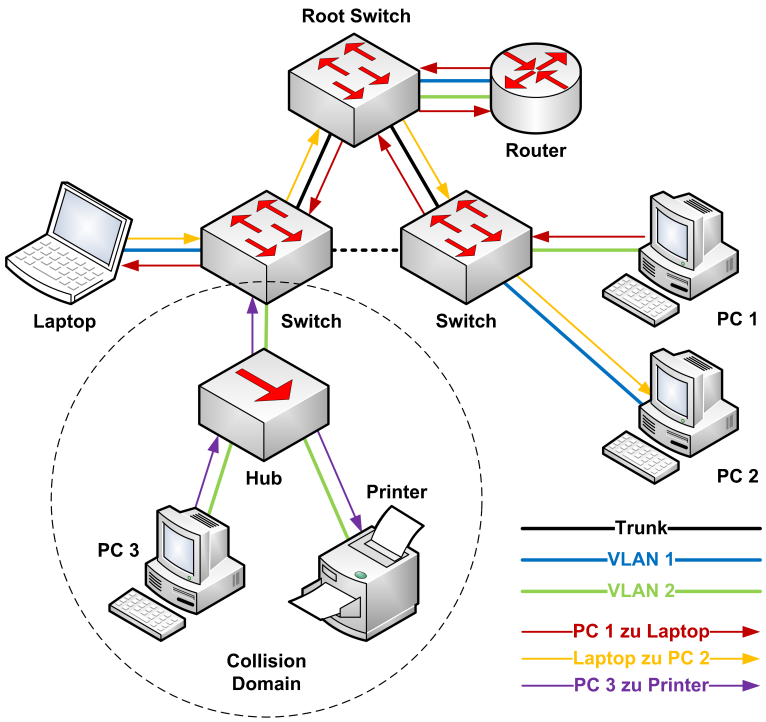


**Abbildung 2.8:** Ethernet mit VLANs

verkehrs) ad absurdum führen würde. Bei dem zuvor genannten Beispiel mit dem IP-Telefon wird diese teilweise Verletzung der Trennung absichtlich in Kauf genommen, etwa wenn, wie erwähnt, nicht genug Netzwerkdosen oder Switch Ports frei sind.

Dennoch kann das Senden von VLAN getaggten Frames beim Discovery hilfreich eingesetzt werden. Normalerweise können von der Discovery-Maschine nur all jene Geräte direkt (d.h. nicht über einen Router) erreicht werden, die sich in der selben Broadcast Domain – was gleichbedeutend mit dem selben VLAN ist – befinden. Kann die Discovery-Maschine jedoch VLAN getaggte Frames senden – was jedoch, wie gesagt, eine entsprechende Konfiguration des Switches, an dem die Discovery-Maschine hängt, voraussetzt –, so kann mit *allen* Geräten im LAN (den Geräten in allen VLANs) direkt kommuniziert werden. Damit sind alle Geräte direkt erreichbar, die sich in der selben „physischen“ Broadcast Domain befinden und die Beschränkung auf die durch das VLAN, in dem sich die Discovery-Maschine befindet, gegebene „virtuelle“ Broadcast Domain fällt weg. Eine damit zusammenhängende Discovery-Technik wird in Abschnitt *3.4.1 ARP Ping* beschrieben.

*Abbildung 2.8* zeigt zusammenfassend ein Ethernet Netzwerk mit VLANs (jedes VLAN entspricht einer Broadcast Domain), wobei einige Kommuni-

kationspfade eingezeichnet sind. So müssen z.B. der `Laptop` und `PC 1` über den Router kommunizieren, da sie zwar in der selben „physischen“ Broadcast Domain, aber in anderen VLANs („virtuellen“ Broadcast Domains) liegen. Für den Datenaustausch über einen Router muss jedoch auf eine höhere Ebene zurückgegriffen werden. Der Router empfängt die Frames in dem einem VLAN und gibt sie im anderen VLAN wieder aus. Zwischen den Switches (über die Trunk Ports) werden Frames aller VLANs, jedoch mit dem entsprechenden VLAN Tag gekennzeichnet, übertragen.

## 2.6 TCP/IP

Während Ethernet heutzutage der de facto Standard auf Layer 2 ist, so ist es *TCP/IP* auf Layer 3/4 und höher. TCP/IP bezeichnet dabei eine gesamte Protokollfamilie, die nach den zwei zentralen Protokollen *TCP* und *IP* benannt ist. TCP/IP wird in diesem Anschnitt näher betrachtet.

In dieser Arbeit wird nur IP in der *Version 4* (*IPv4*) berücksichtigt, da es die momentan am weitesten verbreitetste ist (und es wohl auch noch für einige Zeit bleiben wird), obwohl es schon *IPv6* gibt (siehe [24]).

### 2.6.1 Grundlagen

In einem TCP/IP-Netzwerk erhält jeder Knoten eine eindeutige *IP-Adresse* (eine Netzwerkadresse im Sinne des OSI-Modells), die aus 32 Bit (4 Byte) besteht und meistens in Form von vier durch Punkte getrennte Dezimalzahlen zwischen 0 und 255 (z.B. `212.186.103.151`) dargestellt wird.

Eine IP-Adresse besteht (logisch) aus zwei Teilen – Subnetzadresse (*netid*) und Hostadresse (*hostid*) –, wobei die Länge jedes Teils variabel ist. Je mehr Bits für die Subnetzadresse verwendet werden, desto weniger bleiben für die Hostadresse und umgekehrt. Wie viele Bits pro Adresse verwendet werden bestimmt die *Subnetzmaske* (*Subnet Mask*)[1]. Die Subnetzmaske wird entweder als die Anzahl von für die Subnetzadresse verwendeten Bits, angehängt an die IP-Adresse, angegeben (z.B. `192.168.0.1/24` für eine 24 Bit lange Subnetzadresse) oder in einer Form analog zum Format von IP-Adressen (z.B. `255.255.255.0`). Bei der zweiten Form ist die Maske eine Binärzahl mit der entsprechenden Anzahl von Bits auf `1` gesetzt, gefolgt von den restlichen, auf 32 Bits fehlenden Bits, gesetzt auf `0` (z.B. `11111111 11111111 11111111 00000000`) und dies dann entsprechend den Regeln für eine IP-Adresse dargestellt.

[1] Die Erfinder des IP-Protokolls haben das gesamte weltweite IP-System als ein einziges Netzwerk gesehen, daher ist jede Unterteilung ein „Subnetz“ und kein „Netz“. Subnetze werden verwendet, um die Netzwerklast sinnvoll zu verteilen und Computer nach logischen Kriterien gruppieren zu können.

Durch Vergleich der Subnetzadressen lässt sich feststellen, ob sich ein Empfänger im selben Subnetz wie der Sender befindet. Falls dies so ist, wird die Nachricht direkt gesendet, ansonsten muss sie über Router *geroutet* werden, wie im Abschnitt *2.6.2 Routing* beschrieben wird. Die Subnetzmaske wird, wie erläutert, benötigt, damit man weiß, welcher Teil der IP-Adresse die Subnetzadresse ist, d.h. welche Teile verglichen werden müssen.

Die erste und letzte Adresse in jedem Subnetz haben besondere Bedeutungen. Die erste Adresse (z.B. `212.186.103.0/24`) wird als Bezeichnung für das gesamte Subnetz verwendet. Die letzte Adresse (z.B. `212.186.103.255/24`) ist die sogenannte *Directed Broadcast Address*. Pakete an diese Adresse werden an alle Computer in dem entsprechenden Subnetz weitergeleitet bzw. von diesen verarbeitet. Die übrigen Adressen können frei an Geräte vergeben werden. Beim Discovery ist die Bedeutung dieser besonderen Adressen zu beachten.

Weiters gibt es noch einige spezielle IP-Adressen: Die Adresse `127.0.0.1` (`localhost`) bezeichnet immer den eigenen Computer. Die Adresse `0.0.0.0` bezeichnet eine ungültige bzw. noch nicht gesetzte IP-Adresse. Bei Routern steht diese Adresse auch für die *Default Route*, mehr dazu im Abschnitt *2.6.2 Routing*. Die Adresse `255.255.255.255` ist die sogenannte *Limited Broadcast Address*, damit werden – ähnlich zur Directed Broadcast Address – ebenfalls alle Computer angesprochen, jedoch werden Pakete an diese Adresse von Routern nicht weitergeleitet (daher auch das „limited“ im Namen), da sie sonst an alle Computer im Internet geschickt werden müssten. Der Unterschied zwischen der Directed und Limited Broadcast Address ist der, dass mit der Limited Broadcast Address nur alle Geräte im *gleichen* Subnetz wie der Sender erreicht werden können, während man mit dem Directed Broadcast auch Pakete an alle Geräte eines bestimmten *anderen* Subnetzes schicken kann (dafür muss man aber die Adresse des Subnetzes kennen, während die Limited Broadcast Adresse immer gleich ist). Werden beim Discovery diese speziellen IP-Adressen angetroffen, müssen sie entsprechend berücksichtigt werden.

Da IP-Adressen im Internet immer eindeutig sein müssen, werden von diversen Verwaltungsbehörden Adressbereiche an Firmen, Universitäten etc. vergeben. Wenn jede Organisation nur Adressen aus dem ihr zugeordneten Bereich verwendet, ist sichergestellt, dass keine Adressen doppelt verwendet werden. Da aber einerseits der verfügbare Adressraum nicht ausreicht, um jedem einen Adressbereich zur Verfügung stellen zu können und da es andererseits zu kompliziert wäre, wenn jeder, der TCP/IP verwenden will, zuerst einen Adressbereich beantragen müsste, gibt es sogenannte *private Subnetze* (`10.0.0.0/8`, `172.16.0.0/12` und `192.168.0.0/16`). Adressen aus diesen Bereichen können von Jedermann frei verwendet werden und

können innerhalb der eigenen Organisation auch geroutet werden. Einzig ins Internet dürfen die Adressen nicht geroutet werden (zur Kommunikation mit Geräten im Internet bedarf es in diesem Fall spezieller Techniken, wie in Abschnitt *2.6.2 Routing* beschrieben), da diese Adressen eben nicht eindeutig sind. Es ist auch gebräuchlich, dass diese privaten Adressbereiche beim internen Gebrauch in kleinere Subnetze aufgeteilt werden (z.B. `192.168.0.0/24`, `192.168.1.0/24`, `192.168.2.0/24` usw.). Beim Discovery muss darauf geachtet werden, ob eine Adresse aus einem privaten Bereich stammt oder nicht, denn davon hängt ab, ob sie eindeutig sein muss oder auch mehrfach vorkommen darf.

Da es für Menschen schwierig ist, sich IP-Adressen zu merken, erlaubt das Protokoll *Domain Name System* (*DNS*), welches auf Layer 7 operiert, IP-Adressen (z.B. `74.125.39.106`) Namen (z.B. `www.google.com`) zuzuordnen. Geräte können dann auch über ihren Namen angesprochen werden. Vor der Kommunikation mit einem Gerät, das über seinen Namen identifiziert wird, wird dessen Name in eine IP-Adresse aufgelöst und der Datenaustausch darauf anhand der üblichen Vorgangsweise (als ob die IP-Adresse direkt gegeben gewesen wäre) fortgesetzt. Näheres zu DNS folgt im Abschnitt *2.7.3 Domain Name System (DNS)*.

TCP/IP (siehe [79]) besteht nicht wie das OSI-Modell (siehe [34]) aus sieben, sondern nur aus vier Schichten. Bei TCP/IP fehlen die Layer 5 und 6 (deren Funktionen werden von den darüber- und darunterliegenden Ebenen erfüllt) und die Schichten 1 und 2 werden als eine einzige angesehen.

Auf dem Application Layer (Layer 7) gibt es eine Vielzahl verschiedenster Protokolle, die eine Reihe von Diensten (*Services*) bieten. Einige bekannte sind z.B. *Hypertext Transfer Protocol* (*HTTP*) zur Übertragung von Webseiten, *File Transfer Protocol* (*FTP*) zum Dateitransfer, *Simple Mail Transfer Protocol* (*SMTP*) zum Senden von E-Mails, *Telnet* oder *Secure Shell* (*SSH*) zur Terminal Emulation und viele mehr. Einige für das Discovery besonders relevante Protokolle werden gesondert in Abschnitt *2.7 Weitere Protokolle* behandelt.

Die von den Application Layer Protokollen erzeugten Messages werden an die nächste Schicht, den Transport Layer (Layer 4), zur Übertragung weitergereicht und dort entweder von TCP oder UDP verarbeitet. Näheres dazu in Abschnitt *2.6.3 Datenübertragung*. TCP bzw. UDP verwenden wiederum IP, welches auf dem Network Layer (Layer 3) arbeitet, um die Pakete zum Ziel zu routen. IP bedient sich letztendlich dem Layer 2 Netzwerksystem (z.B. *Ethernet*), um die Daten schließlich zu versenden. Mehr dazu in Abschnitt *2.6.2 Routing*.

Programme kommunizieren bei TCP/IP über so genannte *Sockets* (z.B. `212.186.103.151:80`). Ein Socket ist die Kombination aus der IP-Adresse des Computers, auf dem ein Programm läuft, und des *Ports* (einer einfachen Nummer), der mit diesem Programm assoziiert ist. In diesem Zusammenhang ist mit Port nicht ein Netzwerkinterface gemeint (wie z.B. bei Switches), sondern die Kennung eines Programms auf einem Computer. Hinter jedem Port „verbirgt" sich ein Programm, das die an diesen Port gesandten Daten verarbeitet. Jedem Programm wird eine Port-Nummer zugewiesen, um empfangene Daten an das korrekte Service weiterzuleiten. Damit ist es möglich, verschiedene Programme separat anzusprechen, obwohl sie am selben Computer laufen, daher unter der selben IP-Adresse zu erreichen sind. Stark vereinfacht ausgedrückt könnte man eine IP-Adresse (identifiziert den Computer) mit einer Telefonnummer, die Port-Nummer (identifiziert das Programm) mit einer Nebenstelle vergleichen.

Weit verbreitete Services haben standardisierte (sogenannte *well-known*) Port-Nummern (z.B. HTTP: Port 80, FTP: Port 21, SNMP: Port 161 etc.). Es ist jedoch nicht zwingend erforderlich, ein Programm unter der vorgegebenen Port-Nummer laufen zu lassen, d.h. man kann ein HTTP Service (einen Web Server) unter jedem beliebigen Port laufen lassen (nur müssen Benutzer dieses Services diesen Port dann auch kennen – genau um das zu vereinfachen, gibt es die Standards) bzw. kann man unter Port 80 auch jedes beliebige andere Service laufen lassen. Aus Sicht des Discoverys bedeutet dies nicht, falls ein Gerät auf einem bestimmten Port nicht antwortet, dass das allgemein damit assoziierte Service nicht angeboten wird (es könnte unter einem anderen Port laufen). Umgekehrt heißt es auch nicht, falls eine Antwort von einem bestimmten Port kommt, dass auch das vom Standard her erwartete Service dahinter läuft (um sicherzugehen müsste da erst der Inhalt der Antwort analysiert werden). Dennoch kann man meistens davon ausgehen, dass sich Services an die vorgegebenen Ports halten, da eigene Port-Schemata kompliziert zu verwalten wären.

### 2.6.2 Routing

Das *Internet Protocol* (*IP*, siehe *RFC 791* [64]) arbeitet auf *Layer 3* des OSI-Modells und ist zuständig für das *Routing* von Paketen, d.h. dass sie ihren Weg zum Ziel finden.

Erhält IP ein Paket vom Layer 4, so überprüft es zuerst, ob der Empfänger im selben Subnetz wie der Sender (der Computer auf dem diese Berechnungen stattfinden) liegt. Dazu werden, wie im vorigen Abschnitt beschrieben, die Subnetzadressen des Empfängers und des Senders verglichen, ob sie ident sind. Um zu wissen, welcher Teil der IP-Adressen des Empfängers und Senders die Subnetzadresse ausmacht, dient die Subnetzmaske.

Befinden sich Sender und Empfänger im selben Subnetz, so kann das Paket direkt zum Ziel geschickt werden. Dazu muss als nächstes die Hardwareadresse (meist MAC-Adresse, wenn Ethernet auf Layer 2 verwendet wird) des Empfängers bestimmt werden. Um die der Zieladresse zugeordnete Hardwareadresse zu finden, bedient sich IP des *Address Resolution Protocols* (*ARP*), welches auf Layer 2 operiert und in Abschnitt *2.7.1 Address Resolution Protocol (ARP)* näher beschrieben wird. Sobald die Hardwareadresse des Empfängers bekannt ist, kann IP das Paket in Frames verpacken (wenn ein Paket größer als die maximale Frame-Größe ist, so müssen mehrere Frames pro Paket versandt werden) und diese über Layer 2 direkt an das Ziel schicken. Daraus ergibt sich auch, dass Geräte im selben Subnetz sich auch in der selben Broadcast Domain befinden müssen, ansonsten könnte der Layer 2 die Frames nicht zum Empfänger schicken.

Falls sich Sender und Empfänger in verschiedenen Subnetzen befinden, muss das Paket über einen Router geschickt werden. Entweder ist für das Zielsubnetz im System ein bestimmter Router eingetragen, dann wird das Paket an diesen geschickt, ansonsten wird es an das sogenannte *Default Gateway* gesandt (das ist derjenige Router an den standardmäßig alle Pakete an fremde Subnetze geschickt werden sollen).

Um das Paket an den Router zu schicken, wird – wieder mittels ARP – dessen Hardwareadresse ermittelt und die Frames dann an diese über den Layer 2 geschickt. Daraus ergibt sich, dass sich der Router im selben Subnetz und in der selben Broadcast Domain wie der Sender befinden muss. Daher muss es in jedem Subnetz auch mindestens einen dafür zuständigen Router geben, außer die Geräte im Subnetz sollen nur miteinander und nicht nach außen kommunizieren können.

Ein Router verbindet immer mehrere Subnetze, d.h. er hat Interfaces in verschiedenen Subnetzen. Erhält der Router ein Paket, so führt er den hier beschriebenen Vorgang von Neuem aus. Besitzt der Router ein Interface im Zielsubnetz, so kann er die Hardwareadresse des Empfängers (wieder mittels ARP) bestimmen und die Daten direkt an diesen mit Hilfe des Layer 2 über das entsprechende Interface schicken. Besitzt er kein Interface im Zielsubnetz, so muss er den nächsten Schritt (*Hop*) auf dem Weg zum Ziel bestimmen. Steht der nächste Router (welcher in einem Subnetz liegen muss, mit dem der momentane Router auch verbunden ist) fest, so wird dessen Hardwareadresse bestimmt und die Daten wieder mit Hilfe des Layer 2 über das entsprechende Interface an diesen geschickt.

Dieser Algorithmus wird so lange fortgesetzt, bis ein Router erreicht ist, der auch mit dem Zielsubnetz verbunden ist und der die Daten somit direkt zum Empfänger senden kann oder bis das Paket zu viele Sprünge zwischen

Routern gemacht hat und verworfen wird. Jedes IP-Paket ist mit einem *Time to Live* (*TTL*) Feld versehen, welches von der anfänglichen Sendestation einen Wert (maximal 255) erhält. Bei jedem Router, den das Paket passiert, wird die TTL um eins dekrementiert. Erreicht sie null, so wird das Paket verworfen. Dies dient dazu, damit (z.B. auf Grund von Routing-Fehlern) Pakete nicht endlos im Kreis herumgeschickt werden können.

Eine offene Frage ist nun, wie Router den nächsten, dem Ziel einen Schritt näheren, Router bestimmen. Die Router betrachten dazu nur die Subnetzadresse der Ziel-IP-Adresse und entscheiden anhand dieser sowie interner Tabellen (*Routing Tables*), wohin die Pakete als Nächstes gesandt werden müssen. Diese Tabellen enthalten genau die Information, welcher Router der nächste Schritt auf dem Weg zu einem bestimmten Subnetz ist. Um diese Routing Tables zu generieren und Informationen über die angrenzenden Subnetze und Hosts zu sammeln, verwenden Router eigene Routing-Protokolle (welche auch auf Layer 3 operieren) und tauschen mittels diesen Routing-Daten untereinander aus. Enthält ein Routing Table keinen Eintrag für ein bestimmtes Subnetz, so wird das Paket an den Router, welcher unter der *Default Route* (gekennzeichnet durch das Subnetz `0.0.0.0`) eingetragen ist, geschickt.

In der TCP/IP-Protokollfamilie operiert auf dem Layer 3, neben IP (zur Datenübertragung) und den Routing-Protokollen (zum Aufbau und Austausch der Routing-Informationen in den Routern), auch noch das *Internet Control Message Protocol* (*ICMP*, siehe *RFC 792* [63]), welches dazu dient, den Datentransfer betreffende Steuerungsnachrichten auszutauschen und welches in Abschnitt *2.7.2 Internet Control Message Protocol (ICMP)* näher behandelt wird.

Eine Alternative zu dem zuvor beschriebenen Routing-Algorithmus, stellt das sogenannte *Source Routing* dar. Dabei wird in den IP-Paketen bereits vom Absender die Route zum Ziel eingetragen, anstatt dass die Router jeweils den weiteren Weg bestimmen. Umgekehrt lässt sich bei IP-Paketen zu Diagnosezwecken auch eine *Record Route* Option setzen, womit die Route eines Pakets aufgezeichnet wird, während es die Router passiert. Beides kann im Discovery hilfreich sein, einerseits um Pakete über Routen zu schicken, die sie sonst (nach den Entscheidungen der Router auf dem Weg) nicht nehmen würden und andererseits um Routen aufzuzeichnen und so ein Bild über die Informationen in den Routern zu bekommen. Dennoch werden sowohl Source Routing, als auch Record Route Optionen oft von Routern aus Sicherheitsgründen ignoriert.

Da Routern bei der Datenübertragung eine zentrale Stellung zukommt, vor allem wenn sie als Default Gateway fungieren, haben deren Ausfälle eine

gravierende Wirkung. Fällt z.B. das Default Gateway für ein Subnetz aus, sind alle Geräte in diesem Subnetz von anderen Subnetzen abgeschnitten (innerhalb des Subnetzes können sie weiterhin miteinander kommunizieren) – außer sie haben für manche Zielsubnetze noch explizit andere Router eingetragen. Um sich gegen Hardwareausfälle zu wappnen, wurden verschiedene Protokolle, wie z.B. das *Virtual Router Redundancy Protocol* (*VRRP*, siehe *RFC 3768* [27]) oder das *Hot Standby Router Protocol* (*HSRP*, siehe *RFC 2281* [42]) entwickelt.

Das von der Firma *Cisco* entwickelte, proprietäre HSRP und das Standardbasierende VRRP sind konzeptuell ähnlich, aber technisch miteinander inkompatibel. Bei HSRP/VRRP gibt es *virtuelle Router* (mit dazugehörigen IP- und MAC-Adressen), welche bei den Geräten, z.B. als Default Gateway, eingetragen sind. Hinter jedem virtuellen Router stehen zwei oder mehr physische Router, von denen aber nur einer (der jeweilige *Master*) auf die IP- bzw. MAC-Adresse des virtuellen Routers antwortet. Fällt der Master aus, so übernimmt ein anderer der mit dem virtuellen Router assoziierten physischen Router dessen Funktion. So sieht es für die Endgeräte aus, als ob sie immer mit dem selben Router kommunizieren würden, während sich dahinter mehrere Router verbergen, die sich auch abwechseln können, sollte es zu Ausfällen kommen. Aus Sicht des Discoverys gilt es zwischen virtuellen Routern und den dazugehörigen physischen Routern zu unterscheiden und diese einander zuzuordnen.

Wie im vorigen Abschnitt erwähnt, gibt es private Subnetze, von und zu denen Pakete in das bzw. aus dem Internet nicht geroutet werden können, da die Adressen aus den privaten Subnetzen nicht weltweit eindeutig sind. In solchen Fällen kann *Network Address Translation* (*NAT*, siehe *RFC 3022* [75]) eingesetzt werden. Dabei ist ein privates Subnetz über einen Router (welcher nach außen hin eine gültige, nicht-private, also eindeutige IP-Adresse besitzen muss) mit dem Internet verbunden. Möchte ein Gerät aus dem privaten Subnetz mit dem Internet kommunizieren, schickt es seine Pakete über den Router. Dieser tauscht die (private) Absenderadresse gegen seine eigene (nicht-private) Adresse, womit das Paket weiter im Internet geroutet werden kann. Die Antwort auf dieses Paket wird zurück an die Adresse des Routers geschickt. Dieser muss sich gemerkt haben, zu welchem privaten Gerät dieses Paket gehört (da ja die Antworten auf alle Pakete von allen Geräten aus dem privaten Subnetz an die gleiche Adresse des Routers zurückgeschickt werden), und kann es dann an dieses weiterleiten. Von der Seite des Internets sieht es so aus, als ob es nur den Router gäbe (weil nur seine Adresse auftaucht) und es ist nicht ersichtlich, dass sich dahinter ein privates Subnetz mit mehreren Geräten befindet. Für das Discovery bedeuten NAT-Router ein Hindernis, da mittels Heuristiken maximal erahnt werden kann, dass sich hinter so einem Router eventuell mehrere Geräte ver-

bergen (siehe [6]). Dennoch ist es nicht möglich, die Geräte dahinter direkt zu erreichen und zu analysieren (da sie ja keine eigene, von außen zugängliche IP-Adresse besitzen, unter der sie erreichbar wären).

### 2.6.3 Datenübertragung

Im TCP/IP Stack werden auf *Layer 4* zwei Protokolle eingesetzt (Protokolle höherer Ebenen können sich aussuchen, welches der beiden sie zur Datenübertragung verwenden möchten): das *Transmission Control Protocol* (*TCP*, siehe *RFC 793* [65]) und das *User Datagram Protocol* (*UDP*, siehe *RFC 768* [62]). Zum Versenden ihrer Daten greifen beide auf IP auf Layer 3 zurück.

TCP ist verbindungsorientiert (*connection-oriented*), das heißt, bevor noch irgendwelche Daten gesandt werden können, muss eine Verbindung zwischen den Kommunikationspartnern aufgebaut werden. Ist die Verbindung hergestellt, können die eigentlichen Messages, die von einem höheren Layer stammen, verschickt werden. TCP teilt die Messages dazu in einzelne Pakete auf und hängt an jedes einen TCP-Header an, der u.a. Ziel- und Quell-Ports (zur Kennzeichnung der Services bzw. Programme) sowie Nummern zur Bestätigung (*Acknowledgement Number*) und zur Identifizierung und Einhaltung der richtigen Paketabfolge (*Sequence Number*) enthält. Dann übergibt TCP die Pakete samt Ziel-IP-Adresse an das Protokoll IP auf dem Layer 3 zur eigentlichen Übertragung. Des Weiteren kümmert sich TCP um *Flow Control* (verhindert, dass eine Seite schneller sendet, als die andere empfangen kann) und darum, dass Fehler gefunden und verloren gegangene oder beschädigte Pakete erneut gesandt werden. Da TCP verbindungsorientiert ist, erwartet es nach einer Anzahl von gesandten Paketen eine Bestätigung über deren Erhalt.

Weiters enthält der TCP-Header auch sogenannte *Flags*, das sind einzelne Bits, die auf null oder eins gesetzt werden und damit gewisse Zustände anzeigen können. Dazu gehören u.a. das `ACK` (*Acknowledgement:* Bestätigung), `SYN` (*Synchronization:* z.B. zum Verbindungsaufbau), `RST` (*Reset:* Zurücksetzen der Verbindung) und `FIN` (*Finish:* Beenden der Verbindung) Flag.

Der TCP-Verbindungsaufbau geschieht mit einem *Three-Way Handshake*. Zuerst sendet der *Client* (der die Verbindung aufbauen möchte) an den *Server* (mit dem die Verbindung aufgebaut werden soll) ein TCP-Paket mit einer zufälligen Sequence Number $x$ und gesetztem `SYN` Flag. Dieser antwortet mit einem Paket mit gesetzten `SYN` und `ACK` Flags, gibt als Acknowledgement Number die ursprüngliche Sequence Number plus eins $(x+1)$ an und setzt die eigene Sequence Number wieder auf einen zufälligen

Wert $y$. Hierauf antwortet der Client mit der Acknowledgement Number $y+1$ sowie gesetztem `ACK` Flag. Nach diesem Vorgang, wo beide Seiten eine Bestätigung der Verbindung erhalten haben, ist diese aufgebaut und Daten können übertragen werden.

UDP arbeitet im Gegensatz dazu ohne Verbindungen – es ist *connectionless*. Es müssen keine Verbindungen aufgebaut werden und Daten können „ohne Vorwarnung“ von einer Station zu einer anderen gesandt werden. Auch UDP Pakete erhalten einen Header (u.a. mit den Ziel- und Quell-Ports) und werden zum Versand mit der Zieladresse an IP auf dem Layer 3 weitergereicht.

UDP verwendet weder Flow Control, noch Fehlererkennung oder Fehlerbeseitigung. UDP garantiert nicht, dass eine Message vollständig, korrekt (einzig beschädigte Pakete können erkannt werden, diese werden jedoch ohne Benachrichtigung verworfen) oder in der richtigen Paketreihenfolge den Empfänger erreicht. Sollen diese Eigenschaften sichergestellt werden, so muss sich die höhere Ebene selbst darum kümmern oder TCP zum Transport verwenden. Der Vorteil der Vorgehensweise von UDP ist der geringere Aufwand und damit eine bessere Performance. UDP wird z.B. für realtime Video- und Sprachübertragungen verwendet, wo es keinen Sinn machen würde, verlorene oder beschädigte Pakete nachzusenden, da ihr Inhalt dann sowieso nicht mehr aktuell wäre.

## 2.7 Weitere Protokolle

Dieser Abschnitt behandelt einige weitere Protokolle im Detail, welche eine besondere Relevanz im Discovery besitzen.

### 2.7.1 Address Resolution Protocol (ARP)

Das *Address Resolution Protocol* (*ARP*, siehe *RFC 826* [61]) arbeitet auf *Layer 2* und dient, wie in Abschnitt *2.6.2 Routing* beschrieben, der Zuordnung von MAC- zu IP-Adressen. ARP Nachrichten werden direkt über das zugrunde liegende Netzwerksystem (z.B. Ethernet) versandt.

ARP kann die zu einer IP-Adresse gehörige MAC-Adresse finden. Dazu sendet es einen *ARP Request*, welcher die gesuchte IP-Adresse enthält, an die Broadcast MAC-Adresse (`FF:FF:FF:FF:FF:FF`), sodass alle Geräte in der Broadcast Domain dieses Frame erhalten. Auf dieses Frame antwortet dann dasjenige Gerät mit einem *ARP Reply*, welches die gesuchte IP-Adresse besitzt, inkl. seiner MAC-Adresse. Damit ist dem Sender nun die MAC-Adresse zur gegebenen IP-Adresse bekannt. Es gilt anzumerken, dass die umgekehrte Richtung, das Finden einer IP-Adresse zu einer gegebenen MAC-Adresse, mit ARP *nicht* möglich ist.

Da IP oft viele Pakete pro Sekunde an die gleiche IP-Adresse verschickt, wäre es ineffizient, für jedes dieser Pakete die zur IP-Adresse gehörige MAC-Adresse neu zu bestimmen. Daher wird, wurde einmal die MAC-Adresse zu einer IP-Adresse gefunden, diese Information in einem *ARP Cache*, auch genannt *ARP Table*, gespeichert. Wird eine MAC-Adresse gesucht, wird also zuerst in dieser Tabelle nachgeschaut. Nur wenn diese Tabelle keine entsprechende Information enthält, wird ein ARP Request gesandt. Um veraltete Informationen in den ARP Tables zu vermeiden, werden Einträge gewöhnlich nach 30 Sekunden wieder gelöscht.

Wie zuvor erwähnt, müssen Geräte, die sich im selben Subnetz befinden, auch in der gleichen Broadcast Domain liegen, da sie sonst nicht direkt miteinander kommunizieren können. Sind Geräte des gleichen Subnetzes dennoch in unterschiedlichen Broadcast Domains, so gibt es eine Möglichkeit, wie sie trotzdem Daten austauschen können. Dazu muss ein Router als sogenannter *ARP Proxy* fungieren. Möchte ein Gerät mit einem anderen kommunizieren, das sich im selben Subnetz befindet (dies weiß das erste Gerät anhand der Subnetzadresse in der Ziel-IP-Adresse), aber in einer anderen Broadcast Domain (davon kann das erste Gerät nichts wissen), so sendet es ganz normal einen ARP Request, um die MAC-Adresse des Zielgeräts zu erfahren und seine Daten dorthin zu schicken. Das Zielgerät kann nun aber nicht antworten, da es in einer anderen Broadcast Domain liegt und den ARP Request so gar nicht erst bekommt. Dafür kann aber der ARP Proxy an Stelle des gesuchten Geräts antworten, d.h. er tut so, als ob er das Gerät mit der gesuchten IP-Adresse wäre. Dies veranlasst das erste Gerät seine Daten an den ARP Proxy zu senden, im Glauben es wäre das Zielgerät. Nun kann der ARP Proxy dafür sorgen, dass die Daten zum eigentlichen Zielgerät weitergeleitet werden. Damit ist es z.B. möglich, Netzwerke an zwei entfernten Standorten so zu betreiben, als wären sie ein zusammenhängendes LAN, obwohl dazwischen z.B. eine WAN-Verbindung liegt.

Im Discovery kann ARP auf verschiedenste Weisen – neben der eigentlichen Funktion zum Zuordnen von MAC- zu IP-Adressen – verwendet werden, wie in Abschnitt *3.4 Address Resolution Protocol (ARP)* beschrieben wird. Weiters gilt es besonders ARP Proxies zu berücksichtigen, da deren Verhalten die Ergebnisse verfälschen kann (inkorrekte IP-MAC-Zuordnungen).

### 2.7.2 Internet Control Message Protocol (ICMP)

Das *Internet Control Message Protocol* (*ICMP*, siehe *RFC 792* [63]) arbeitet auf *Layer 3*, ist ein integraler Bestandteil von IP und wird von diesem zu Diagnose- und Steuerungszwecken verwendet. Dazu können eine Reihe von *Control Messages* verschickt werden. Zum eigentlichen Versand dieser Nachrichten wird IP verwendet.

Zu diesen Nachrichten gehören u.a. `Destination Host Unreachable` (geschickt von Routern, wenn das Zielgerät nicht erreichbar ist, z.B. niemand auf den ARP Request des Routers im Zielsubnetz antwortet), `Destination Port Unreachable` (geschickt von Geräten, die eine Nachricht an einen Port bekommen haben, unter dem kein Service läuft), `Time Exceeded` (geschickt von Routern, wenn ein Paket verworfen wurde, weil seine TTL abgelaufen ist) und viele mehr.

Weitere bekannte Nachrichten sind `Echo Request` und `Echo Reply`, welche vom Programm `Ping` (siehe *RFC 2151* [38]) verwendet werden. Dieses dient dazu, um festzustellen, ob ein Zielgerät aktiv und die Verbindung zu ihm intakt ist. Dazu wird eine Echo Request Nachricht mit beliebigen Daten an das Gerät geschickt, welches mit einer Echo Reply Nachricht antwortet, welches die erhaltenen Daten zurückschickt. Wird ein Echo Reply mit den ursprünglich versendeten Daten empfangen, so bedeutet das, dass das Zielgerät vorhanden und die Verbindung in Ordnung ist. Zusätzlich kann z.B. die Zeit gemessen werden, die für die Antwort gebraucht wurde.

Auch ICMP kann im Discovery auf verschiedene Weisen eingesetzt werden, wie in Abschnitt *3.6 Internet Control Message Protocol (ICMP)* beschrieben.

### 2.7.3 Domain Name System (DNS)

Das *Domain Name System* (*DNS*, siehe *RFC 1034* [55] und *RFC 1035* [56]) arbeitet auf *Layer 7* und dient, wie in Abschnitt *2.6.2 Routing* beschrieben, der Zuordnung von *Netzwerknamen* zu IP-Adressen. DNS Nachrichten werden über UDP versandt.

Um IP-Adressen Namen zuordnen zu können (und umgekehrt), gibt es zwei grundlegende Operationen: *Lookup* (sucht die IP-Adresse zu einem Namen) und *Reverse Lookup* (sucht den Namen zu einer IP-Adresse). Diese Informationen werden von DNS Servern abgefragt, welche Tabellen mit Einträgen für beide Richtungen besitzen (also für ein bestimmtes IP-Name-Paar einen Eintrag, wo der Name auf die IP-Adresse verweist und einen anderen, wo die IP-Adresse auf den Namen verweist). Weiters gibt es noch verschiedene andere Arten von Einträgen, mit denen weitere Informationen gespeichert und abgefragt werden können (z.B. welches Gerät der Mail oder Web Server für eine Domain ist usw.). Für Details dazu sei auf *RFC 1035* [56] verwiesen.

Im DNS können auch *Alias-Namen* verwaltet werden. So kann z.B. die IP-Adresse `10.20.30.40` dem Namen `www.xyz.com` zugeordnet sein und dieser Name gleichzeitig den Alias `alias.xyz.com` besitzen. In diesem Fall sind im DNS drei zugehörige Einträge vorhanden: beim ersten Eintrag verweist die IP-Adresse auf den Namen, beim zweiten Eintrag verweist der Name auf die IP-Adresse, beim dritten Eintrag verweist der Alias-Name auf den Namen.

Wird ein Name zur IP-Adresse gesucht, wird der erste Eintrag verwendet. Wird eine IP-Adresse zum Namen gesucht, wird der zweite Eintrag verwendet. Wird eine IP-Adresse zum Alias-Namen gesucht, wird zuerst der dritte Eintrag verwendet, um den eigentlichen Namen zum Alias zu finden und dann der zweite Eintrag, um die IP-Adresse zu diesem Namen zu finden. Eine Konsequenz dieser Funktionsweise ist jedoch, dass zu einer gegebenen IP-Adresse nur der eigentliche, aber nie der Alias-Name gefunden werden kann, weil es keinen Eintrag gibt, der von der IP-Adresse auf den Alias-Namen (oder vom Namen auf den Alias-Namen) verweist. Daher ist beim Discovery zu bedenken, dass zu einer IP-Adresse nur ein Name und nicht alle vorhandenen Namen gefunden werden können (auch wenn umgekehrt von allen Namen auf die IP-Adresse geschlossen werden kann).

Jeder DNS Server verwaltet einen bestimmten Bereich, genannt *Zone*, und enthält Informationen über diesen. Wenn ein DNS Server all seine Informationen an einen anderen überträgt (z.B. weil der zweite Server ein Backup-Server ist, der die Funktion des ersten übernimmt, falls dieser ausfällt), wird dies *Zone Transfer* genannt.

Im Discovery kann DNS, neben der Zuordnung von IP-Adressen zu Namen, auch dazu dienen, Hinweise auf möglich vorhandene Geräte zu erhalten, was näher im Abschnitt *3.9 Domain Name System (DNS)* behandelt wird.

### 2.7.4 Simple Network Management Protocol (SNMP)

Das *Simple Network Management Protocol* (*SNMP*, siehe *RFC 1157* [12]) arbeitet auf *Layer 7* und dient der Überwachung und Verwaltung von Geräten. SNMP Nachrichten werden ebenfalls über UDP versandt. Der Inhalt der Nachrichten wird mit ASN.1 kodiert. SNMP gibt es in verschiedenen Versionen (*SNMPv1*, *SNMPv2* und *SNMPv3*), diese Arbeit beschränkt sich auf die Betrachtung und Verwendung von Version 1, da diese die simpelste ist und so der Fokus auf dem eigentlichen Discovery liegen kann und nicht zusätzliche Interna des SNMP-Protokolls behandelt werden müssen.

Geräte, die über SNMP verwaltet werden, stellen Informationen zur Verfügung, die ausgelesen und/oder geschrieben werden können. Damit auf diese Informationen zugegriffen werden kann, muss in SNMP Nachrichten eine Art Passwort mitgeschickt werden (welches im Klartext übertragen wird), von denen es in SNMPv1 zwei gibt: die *Read Community* (wird benötigt zum Auslesen von Daten) und die *Write Community* (wird benötigt zum Schreiben von Daten). Nur wenn ein Gerät eine SNMP Nachricht mit korrekter Community (wie sie zuvor auf dem Gerät definiert wurde) empfängt, wird die Nachricht auch beachtet. Höhere Versionen von SNMP bieten auch ausgefeiltere und sicherere Methoden zur Authentifizierung.

Weiters enthält jedes SNMP Paket eine sogenannte *Request ID*, um Antworten den entsprechenden Anfragen zuordnen zu können.

Die Informationen in einem SNMP-Gerät sind in Form eines Baumes strukturiert, wobei sich hinter jedem Knoten des Baumes ein (skalarer) Wert verbirgt bzw. verbergen kann. Um einen bestimmten Wert auszulesen (oder zu überschreiben), muss dessen Position im Baum angegeben werden, was über einen *Object Identifier* (*OID*) geschieht. So gibt z.B. die OID `1.3.6.1.2.1.1.5.0` den ersten Knoten unter der Wurzel, dann dessen drittes Kind, dann wiederum dessen sechstes Kind usw. an. Die Null am Ende bedeutet, dass damit der eigentliche Wert des zuvor angesprochene Knoten selbst und nicht eines seiner Kinder gemeint ist.

Es reicht jedoch nicht, einfach nur den Wert eines Knotens im Baum auszulesen oder zu setzen, es muss auch gewusst werden, wofür dieser Wert steht. Dafür gibt es die sogenannten *Management Information Bases* (*MIB*), die definieren, welche Information sich hinter welchem Knoten verbirgt. Je nachdem, welche MIBs einzelne Geräte unterstützen, weiß man, welche Informationen dieses Gerät liefern kann. Verschiedenste Arten von Geräten implementieren verschiedenste MIBs (so macht es z.B. kaum Sinn, dass die `PRINTER-MIB`, welche u.a. Informationen über verbleibendes Papier und Tinte anbietet, von anderen Geräten, als Druckern unterstützt wird).

Eine von praktisch allen Geräten angebotene MIB ist z.B. die `SNMPv2-MIB` (auch wenn diese MIB SNMPv2 im Namen hat, so kann sie dennoch auch über SNMPv1 abgefragt werden). Diese definiert unter anderem, dass sich hinter der OID `1.3.6.1.2.1.1.5.0` (wofür in der MIB zusätzlich die Abkürzung `sysName` definiert ist) der Name des Geräts verbirgt. Möchte man also wissen, wie ein Gerät heißt, so muss man diese OID auslesen. Die Zeit, wie lange das Gerät schon eingeschaltet ist, findet man unter `1.3.6.1.2.1.1.3.0` (`sysUpTime`) usw.

Über SNMP können auch Tabellen ausgelesen werden, wobei diese jedoch in die Baumstruktur passen müssen und dadurch die Vorgehensweise etwas unintuitiv wird. Das `ifTable` zum Beispiel enthält Informationen über die Interfaces eines Geräts. Diese Tabelle definiert mehrere Spalten, unter anderem für den Index des Interfaces (`ifIndex`) und eine textuelle Beschreibung des Interfaces (`ifDescr`). Die Spalte `ifIndex` hat die OID `1.3.6.1.2.1.2.2.1.1`, der Wert der ersten Zeile in dieser Spalte wäre unter `1.3.6.1.2.1.2.2.1.1.1`, der der zweiten Zeile unter `1.3.6.1.2.1.2.2.1.1.2` usw. zu finden. Analog dazu hat die Spalte `ifDescr` die OID `1.3.6.1.2.1.2.2.1.2`. Der Wert in der ersten Zeile dieser Spalte wäre dann unter `1.3.6.1.2.1.2.2.1.2.1`, der der zweiten Zeile unter `1.3.6.1.2.1.2.2.1.2.2` usw. zu finden. An die OID der Spalte wird

also der Index der Zeile angehängt. Es ist anzumerken, dass der Zeilenindex nicht immer kontinuierlich verlaufen muss, sondern es kann auch nach anderen Kriterien indiziert werden. So gibt es z.B. Tabellen, die nach einer IP-Adresse indiziert sind. Würde sich eine Zeile z.B. auf die IP-Adresse `212.186.103.151` beziehen, so wäre der Index dieser Zeile ebenfalls `.212.186.103.151`, welcher an die OID der gewünschten Spalte angehängt werden muss (bzw. kann man sich mittels `GET-NEXT` Abfragen von einer Zeile zur nächsten vorarbeiten, sodass der Zeilenindex nicht explizit angegeben werden muss). Das korrekte Auslesen von Tabellen kann jedoch Software-Tools und -Bibliotheken überlassen werden. Wichtig ist nur die OID der Werte und Spalten zu kennen, die man auslesen möchte.

SNMP spielt beim Discovery eine zentrale Rolle, weil es eine standardisierte Möglichkeit bietet, um Informationen von den Geräten selbst abzufragen. Besonders interessant sind dabei Informationen von Infrastrukturkomponenten (wie Router und Switches), da diese, bedingt durch ihre Funktion, viel über das Netzwerk wissen. Näher wird dieses Thema in Abschnitt *3.3 Simple Network Management Protocol (SNMP)* behandelt.

# Kapitel 3

# Grundlagen des Network Discoverys

Aufbauend auf den im vorigen Kapitel vorgestellten Grundlagen von Computernetzwerken, kann nun begonnen werden, das eigentliche Thema dieser Arbeit – das Network Discovery – zu behandeln.

## 3.1 Grundlagen

Als Erstes gilt es den Begriff *Network Discovery* zu definieren, wie er in dieser Arbeit verstanden und verwendet wird. Im Kontext dieser Arbeit bedeutet *Network Discovery* das *automatische Finden von Informationen über ein Netzwerk (LAN)*. Dies beinhaltet sowohl Daten über dessen (Infra-)Struktur (Komponenten, Topologie) als auch über daran angeschlossene Geräte (Typen, Adressen, Services etc.). Es geht also um das Finden von Geräten in einem Netzwerk sowie von Informationen über diese und deren Verbindungen untereinander. Konkret werden in diesem Kapitel Techniken zur Erfassung der folgenden Daten behandelt:

- Layer 1/2 Infrastruktur (z.B. Hubs, Switches, VLANs) und Topologie
- Layer 3 Infrastruktur (z.B. Router, Subnetze) und Topologie
- Angeschlossene Geräte
- Geräteeigenschaften (z.B. IP- und MAC-Adressen, Namen, Services)

Die im Folgenden vorgestellten Methoden liefern Daten zu jeweils einem oder mehreren dieser Punkte. In diesem Kapitel werden mögliche Techniken – inklusive deren Eigenschaften sowie Vor- und Nachteilen –, gruppiert nach dem zugrundeliegenden Protokoll, angeführt und erklärt (sowie

im Abschnitt *3.12 Leitfaden* nochmals überblicksartig betrachtet). Im darauffolgenden Kapitel *4 Der Discovery-Algorithmus* erfolgt anschließend eine Auswahl und Zusammenfassung von Methoden zu einem Algorithmus.

Neben der Definition von Network Discovery gilt es auch, diesen Begriff in einen größeren Kontext einzuordnen. Das Network Discovery kann dem Gebiet des *Network Managements* zugerechnet werden. Das sind alle Aktivitäten, die sich unter folgenden Punkten einordnen lassen:

- *Operation:* Betrieb und Aufrechterhaltung der Funktion des Netzwerkes. Dazu gehört auch dessen Überwachung (*Network Monitoring*), um Fehlerzustände möglichst schnell erkennen und entsprechend handeln zu können.
- *Administration:* Verwaltung der Ressourcen und deren Verwendung im Netzwerk. Dazu gehört u.a. das *Asset Management* (siehe [39]).
- *Maintenance:* Wartung und Reparaturen im Netzwerk. Dazu gehört das Beheben von Fehlern genauso wie Präventivmaßnahmen.

Eine Art, die Funktionen des Network Managements zu charakterisieren, bietet das in [16] beschriebene *FCAPS*-Modell der ISO:

- *Fault Management:* Fehlerzustände im Netzwerk erkennen, aufzeichnen und beheben.
- *Configuration Management:* Verwaltung der Konfigurationen von Netzwerk-Ressourcen und -Komponenten.
- *Accounting Management:* Überwachung der Nutzung von Netzwerk-Ressourcen, um Quoten zu kontrollieren und/oder Kosten abzurechnen.
- *Performance Management:* Leistungsdaten messen, analysieren und überwachen, um eine ausreichend performante Funktion der Netzwerk-Ressourcen sicherzustellen.
- *Security Management:* Beschränkung des Zugriffs auf Netzwerk-Ressourcen, um vertrauliche Informationen zu schützen sowie unberechtigte Nutzung oder Manipulationen zu verhindern.

Die Frage ist nun, in welchem Bezug das Network Discovery zu all den genannten Tätigkeiten und Aufgaben steht. Den angeführten Punkten ist gemein, dass sie alle auf einem Wissen über die Struktur des Netzwerkes und die darin vorhandenen Komponenten aufbauen. Beim *Network Monitoring* muss etwa zuerst bekannt sein, welche Geräte es überhaupt zum Überwachen

gibt. Analoges gilt beispielsweise für das *Asset Management* (Inventarisierung von Geräten) und das *Configuration Management*, wo zuerst die Information notwendig ist, welche Geräte zum Inventarisieren bzw. Konfigurieren vorhanden sind. An dieser Stelle fügt sich das Network Discovery ein, welches genau diese Daten liefern kann.

Jetzt könnte eingewendet werden, dass ein Netzwerkmanager sein Netzwerk kennen und auch wissen muss, welche Komponenten er angeschafft hat und sich somit im Netzwerk befinden. Daher sei es gar nicht nötig, diese automatisch erfassen zu wollen. Dem kann jedoch entgegnet werden, dass es einerseits durch die Fülle von Daten (an ein modernes LAN können tausende von Geräten angeschlossen sein) nicht möglich ist, ohne irgendeiner Art von Dokumentation die Übersicht zu behalten. Andererseits kommt hinzu, dass durch die Menge an Detailinformationen, die sich schnell und auch unbemerkt ändern können, eine händisch geführte Dokumentation bei einem Netzwerk einer gewissen Größenordnung praktisch nicht mehr ständig aktuell gehalten werden kann, gar nicht zu sprechen von der menschlichen Fehleranfälligkeit. Dies kann umgangen werden, indem ein System eingesetzt wird, welches die nötigen Daten automatisch bestimmen kann.

Genauso kann auch der Fall eintreten, dass etwa das Management eines Netzwerkes an eine andere Person oder Organisation übertragen wird, wobei keine ausreichende Dokumentation des Netzes vorhanden ist. In dieser Situation können erste Informationen zur Orientierung – sowie in weiterer Folge im Detail etwa zur Konfiguration von Network Monitoring Systemen usw. – ebenfalls durch das Network Discovery gewonnen werden.

Die praktische Relevanz und der Nutzen der vom Network Discovery gesammelten Daten wurden bereits im Abschnitt *1.1 Problemstellung* behandelt. In Kapitel *7 Einsatzszenarien* werden weitere Situationen vorgestellt, in denen das Network Discovery beim Verwalten eines Netzwerkes helfen kann.

## 3.2 Ansätze

Es gibt mehrere konzeptuelle Ansätze – jeweils mit Vor- und Nachteilen –, wie bestimmte Methoden an das Problem des Network Discoverys herangehen bzw. nach denen die Techniken klassifiziert werden können, welche in diesem Abschnitt betrachtet werden. Aus diesen möglichen Ansätzen ergeben sich auch die grundlegenden Designentscheidungen bei der Entwicklung eines Discovery-Algorithmus'. Natürlich können in einem System oder Algorithmus auch Methoden, die unterschiedlichen Ansätzen folgen, kombiniert werden – dies ist sogar ratsam, um das volle Potential des Discoverys ausschöpfen zu können.

### 3.2.1 Aktiv vs. passiv

Die erste Unterscheidung ergibt sich dadurch, ob eine Discovery-Methode *aktiv* oder *passiv* arbeitet. Aktiv bedeutet, dass sie zum Discovery selbst Daten erzeugt und über das Netzwerk versendet, während passive Techniken nur bestehenden Netzwerk-Traffic (z.B. unter Verwendung des Promiscuous Modes eines Netzwerkadapters) abhören.

Der Vorteil des zweiten Ansatzes ist die Tatsache, dass das Netzwerk durch das Discovery nicht nur nicht belastet wird (auch wenn dieses Kriterium auf Grund der Leistungsfähigkeit moderner Netze kaum mehr ins Gewicht fällt), sondern im Netzwerk auch gar nicht bemerkt werden kann, dass überhaupt ein Discovery stattfindet. Dafür hat man aber keine Kontrolle, welche und wie viele Informationen gerade über das Netzwerk gesendet und somit abgehört werden können, während bei aktiven Methoden gezielt nach der im Moment gewünschten Information gesucht werden kann. Daher benötigt passives Discovery auch eher eine länger andauernde Überwachung. Umgekehrt können aber unter Umständen nicht alle Informationen auch aktiv erfragt werden. So werden z.B. Clients mit einem File-Server oder Router untereinander über die entsprechenden Protokolle kommunizieren (was abgehört werden kann), sich aber weigern, über diese Protokolle auch direkt mit der Discovery-Maschine Daten auszutauschen. Router kennen z.B. ihre jeweiligen Nachbarn und kommunizieren über spezielle Routing-Protokolle aus Sicherheitsgründen nur untereinander.

Weiters können immer nur Daten aus der selben Collision Domain mitgehört werden, wodurch der passive Ansatz bei heutigen, meist vollgeswitchten Netzwerken an Wirksamkeit verliert und nur in Spezialfällen seine Bedeutung behält. In so einem Fall müsste die Discovery-Maschine an einem speziellen Diagnose-Port eines Switches (z.B. in der „Nähe“ eines Routers, falls Routing-Protokolle abgehört werden sollen) angeschlossen werden, der explizit dafür konfiguriert ist, allen Traffic weiterzuleiten. Das erfordert aber physischen und administrativen Zugang zu diesem Switch. Auf Grund der Funktionsweise des Switchings kann aber nicht garantiert werden, dass jeglicher Traffic überhaupt diesen erreicht. Um dies völlig zu umgehen, müsste an jedem Switch eine eigene Discovery-Maschine hängen.

Auf Grund der zuvor genannten Einschränkungen sind die meisten der heutzutage eingesetzten Methoden aktiv. Einige passive Methoden werden beispielsweise in [86] und [57] vorgestellt. Für passive Techniken gilt, dass generell jeglicher Traffic mit einer geeigneten Methode abgehört werden kann. Je nachdem, von welchem Protokoll Daten empfangen werden, kann man daraus Informationen beziehen, die über dieses transportiert werden. Aus jeder Kommunikation lassen sich im Allgemeinen Sender und Empfänger extrahieren, was zumindest auf deren Existenz im Netzwerk hinweist. Bei passi-

ven Methoden liegt der Fokus mehr auf der Analyse der abgehörten Daten eines bestimmten Protokolls – wofür sich als Hilfestellung die Spezifikation des jeweiligen Protokolls am besten eignet – als auf bestimmten Schritten, die in einer gewissen Reihenfolge durchgeführt werden müssten, um an die gewünschten Informationen zu gelangen. Dies ist ein Grund (neben der eingeschränkten Wirksamkeit in modernen Netzwerken), warum dieser Ansatz in dieser Arbeit nicht weiter verfolgt wird.

### 3.2.2 Probing vs. Polling

Man kann unterscheiden, ob eine Methode Pakete aussendet, um mit deren Hilfe (d.h. den Reaktionen darauf) Informationen über das Netzwerk zu sammeln (*Probing*) oder die Technik selbst keine Daten gewinnt, sondern diese nur von einer anderen Komponente im Netzwerk ausliest, welche die gewünschten Daten besitzt (*Polling*). Beide Ansätze sind aktiv, da dafür explizit Daten über das Netzwerk gesendet bzw. ausgetauscht werden.

Zum ersten Ansatz gehören etwa das Senden von ICMP Echo Paketen (siehe *3.6.1 Ping*) oder der Traceroute-Algorithmus (siehe *3.6.4 Traceroute*). Je nachdem ob und welche Antworten auf diese Probe Packets kommen, lassen sich Rückschlüsse auf aktive Geräte bzw. die Routing-Struktur ziehen. Beim zweiten Ansatz hingegen werden die gesuchten Daten (z.B. die Routing Tables) direkt von einer Komponente (z.B. ein Router) ausgelesen (meist über SNMP), welche die gesuchten Informationen kennt.

Der Vorteil dieses zweiten Ansatzes liegt darin, dass viele Geräte (meist Infrastruktur-Komponenten) bereits eine Vielzahl an Informationen über das Netzwerk besitzen und auf Grund ihrer Funktion auch besitzen müssen, die nur abgefragt werden müssen. Dazu kommt, dass es oft auch gar keine andere Möglichkeit gibt, an gewisse Daten zu gelangen, als sie von einem bestimmten Gerät auszulesen.

Umgekehrt müssen einerseits die abzufragenden Komponenten aber auch gewillt sein, ihre Informationen an die Discovery-Maschine weiterzugeben (d.h. Passwörter und dergleichen müssen bekannt sein), andererseits auch mit der entsprechenden Methode zum Auslesen der Daten (z.B. SNMP) ansprechbar sein. Dies ist beispielsweise bei Routern oder Switches meist, bei Druckern und Servern manchmal und bei Workstations und Laptops selten der Fall. Alles immer unter der Voraussetzung, dass die gefragte Komponente die gewünschten Daten überhaupt kennt. Weiters kommt hinzu, dass je nach Hersteller und Typ des jeweiligen Gerätes die Daten oft in verschiedenen Formaten (weil etwa unterschiedliche MIBs unterstützt werden) oder Qualitäten vorliegen (selbst wenn Variablen oder Tabellen dem gleichen Format folgen, so können sie dennoch mit unterschiedlichen Werten befüllt

sein). Dies erschwert die Entwicklung einer Methode, die unabhängig von einem bestimmten Hersteller oder Gerätetyp funktioniert.

Beim Probing kann es zwar auch zu unterschiedlichem Verhalten von Geräten kommen, etwa ob und wie sie auf ein Probe Packet antworten. Dennoch kann hier im Vergleich zum Polling meist mit einer einheitlichen Methode vorgegangen werden. Weiters ist das Probing nicht davon abhängig, ob ein bestimmtes Gerät gewisse Informationen kennt und auch zur Verfügung stellt, sondern kann unabhängig von anderen Komponenten im Netzwerk eingesetzt werden, da keine Daten abgefragt, sondern die Reaktionen auf Probe Packets beobachtet werden.

Probing (z.B. Suche nach aktiven Geräten) und Polling (z.B. Abfrage von Spanning Tree Daten) ergänzen einander gut, da manche Informationen nur mit dem einen, manch andere nur mit den anderen Ansatz bestimmt werden können. Daher werden in dieser Arbeit beide Ansätze verwendet und kombiniert.

### 3.2.3 Zentral vs. verteilt

Beim Discovery kann weiters unterschieden werden, ob es von einer Stelle aus (*zentral*) oder von mehreren Punkten (*verteilt*) durchgeführt wird. Zentrale Methoden sind meist weniger komplex, da keine Koordination zwischen den einzelnen Discovery-Maschinen und keine Konsolidierung der von ihnen gesammelten Daten nötig ist. Hingegen können mit einem verteilten Discovery meist mehr Informationen bestimmt werden – vor allem dann, wenn der Ort, von dem aus das Discovery durchgeführt wird, eine Rolle spielt und nicht alle Teile des Netzwerkes von überall aus „eingesehen“ werden können. Oft können Methoden, die im Allgemeinen von einer Stelle aus ausgeführt werden, auch von mehreren Punkten aus angewendet werden, um damit an mehr bzw. bessere Daten zu gelangen. In diesem Fall sind die Techniken nicht als per se verteilt anzusehen, sondern sie werden einfach mehrfach von verschiedenen Orten aus eingesetzt. Umgekehrt gibt es aber auch Methoden, die explizit darauf angewiesen sind, an verschiedenen Stellen Daten zu senden und zu empfangen (siehe z.B. [8]).

Verteiltes Discovery ist vor allem bei passiven Methoden relevant, da es hier besonders auf die Stelle ankommt, an der Daten abgehört werden. Es kann jedoch auch bei aktiven Methoden hilfreich sein, etwa wenn Firewalls gewisse Pfade im Netzwerk blockieren. Meist macht verteiltes Discovery nur beim Probing Sinn, da beim Polling Geräte gewöhnlicherweise immer die gleichen Daten liefern, egal von wo und wem sie gefragt wurden (solange der Fragesteller die entsprechenden Rechte besitzt). Dennoch kann es auch hier der Fall sein, dass gewisse Komponenten nur Anfragen, die von bestimmten

IP-Adressen oder Subnetzen kommen, beantworten bzw. dass die Kommunikation zur Datenabfrage nicht über alle Wege möglich ist (z.B. ebenfalls auf Grund von Firewalls, die etwa SNMP-Traffic blockieren).

In dieser Arbeit werden Methoden vorgestellt, die an sich von einem einzelnen Punkt aus eingesetzt funktionieren (unter der Annahme, dass die gesendeten Daten nicht blockiert werden), aber auch an mehreren Stellen gleichzeitig eingesetzt werden können, um mehr bzw. bessere Daten zu erhalten. In diesem Fall bleibt es jedoch dem Leser überlassen, wie die Daten von verschiedenen Discovery-Maschinen konsolidiert und mit etwaigen Diskrepanzen umgegangen werden soll.

### 3.2.4 Network vs. Device Assistance

Abschließend kann unterschieden werden, ob einerseits das Discovery Eigenschaften des Netzwerkes und der Komponenten darin ausnützt (z.B. welchen Weg ein Traceroute-Paket nimmt, welche Geräte auf Ping antworten) bzw. sich Informationen von gewissen Geräten (z.B. über SNMP ausgelesene Routing Tables) zu Nutze macht (*Network Assistance*) oder ob andererseits alle Geräte von sich aus Pakete senden, die Daten über sie selbst enthalten oder mit Hilfe derer auf die Struktur des Netzwerkes geschlossen werden kann (*Device Assistance*).

Im zweiten Fall muss auf einigen oder allen Geräten im Netzwerk ein Programm (*Daemon*) laufen, welches Daten (z.B. an die Discovery-Maschine oder an andere Geräte) ausschickt. Diese Pakete können einerseits Informationen über das Gerät enthalten, welches die Daten schickt. Andererseits kann auch über den Weg, den verschiedene Pakete nehmen, auf die Struktur des Netzwerkes geschlossen werden. Ein Algorithmus dazu, der ein proprietäres, eigens dazu entwickeltes Protokoll verwendet, wird in [8] beschrieben.

Der Vorteil dieser Vorgehensweise besteht darin, dass ein beliebiges eigenes Protokoll mit einem dazugehörigen Algorithmus entwickelt werden kann, ohne etwa auf Einschränkungen bestehender Protokolle oder Datenformate von Komponenten unterschiedlicher Hersteller etc. Rücksicht nehmen zu müssen. Jedoch ist von Nachteil, dass der entsprechende Daemon (das Programm, welches das Protokoll implementiert) auf allen gewünschten Geräten installiert sein muss. Dies mag in einer großen Organisation, bei der die Geräte jeweils nach einem festgelegten Standard aufgesetzt sind, mit vertretbarem Aufwand möglich sein, in vielen anderen Fällen gibt es aber z.B. keinen Zugriff auf die Rechner der Benutzer. Außerdem gibt es Arten von Geräten (wie etwa Switches, Router oder Drucker) auf denen nicht einfach irgendeine Software installiert werden kann.

In dieser Arbeit wird dieser Ansatz deswegen nicht weiter verfolgt, da normalerweise nicht davon ausgegangen werden kann, dass eine Discovery Software überall im Netz installiert werden kann.

## 3.3 Simple Network Management Protocol (SNMP)

SNMP, dessen Funktionsweise bereits in Abschnitt *2.7.4 Simple Network Management Protocol (SNMP)* beschrieben wurde, nimmt eine zentrale Stellung im Discovery ein, da damit eine Vielzahl von Informationen von anderen Geräten (vor allem Infrastrukturkomponenten) abgefragt werden kann und nicht selbst bestimmt werden muss (was oft auch gar nicht möglich wäre, wie z.B. bei Spanning Tree Daten, die nur Switches bekannt sind).

Damit SNMP erfolgreich im Discovery eingesetzt werden kann, müssen einige Voraussetzungen erfüllt sein. Die Geräte, von denen Informationen ausgelesen werden sollen, müssen SNMP prinzipiell unterstützen, SNMP muss auf ihnen auch aktiviert sein und die erforderlichen SNMP Communities zum Zugriff müssen bekannt sein.

Ein Umstand, der die Arbeit mit SNMP erschwert, ist, dass das Verhalten von SNMP-Geräten oft stark von deren Typ und Hersteller abhängt. Einerseits gibt es in vielen Fällen mehrere (standardisierte oder proprietäre) MIBs, die die gleichen oder ähnliche Informationen anbieten. Je nachdem, welche dieser MIBs von einem Gerät angeboten werden, muss unterschiedlich damit umgegangen werden, sodass ein einheitliches Vorgehen unmöglich wird. Andererseits kann es auch sein, dass selbst in dem Fall, dass verschiedene Geräte die gleiche MIB unterstützen, es dennoch Unterschiede in den darin angebotenen Daten geben kann. Das muss wiederum berücksichtigt werden, auch wenn die Daten im zumindest gleichen Format vorliegen.

Es ist zu beachten, dass auf einem Gerät mehrere *SNMP Engines* bzw. *SNMP Agents* laufen können, die über verschiedene Communities angesprochen werden. Das heißt, dass von einer IP-Adresse verschiedene Informationen geliefert werden können, je nachdem mit welcher Community diese angefragt werden. Dies wird beispielsweise bei manchen Geräten eingesetzt, die gleichzeitig Switching- und Routing-Funktionen besitzen – unter einer Community stellen sich diese als Switches, unter einer anderen als Router dar. Beim Discovery muss dies entsprechend berücksichtigt werden, dass es sich in solchen Fällen um jeweils ein Gerät mit mehreren Aufgaben handelt.

Weiters wird diese Technik z.B. bei *Cisco*-Geräten zum sogenannten *Community Indexing* eingesetzt. Dabei wird die ID eines VLANs an die Commu-

nity angehängt (getrennt durch einen Klammeraffen, z.B. `public@1`). Dies bewirkt, dass das Gerät dann nur Informationen liefert, die zum angegebenen VLAN gehören. Wird beispielsweise ein AFT ausgelesen, so sind darin nur MAC-Adressen und Ports enthalten, die im definierten VLAN liegen.

Ein Gerät muss aber nicht zwingend verschiedene Informationen unter verschiedenen Communities liefern. Manche Geräte sind über mehrere Communities ansprechbar, liefern aber überall die gleichen Daten. Auch dieser Fall muss entsprechend berücksichtigt werden. Generell kann sich das Verhalten von Geräten in diesem Bezug stark unterscheiden. Manche Geräte antworten beispielsweise auf jede beliebige Community, andere antworten mit einer anderen Community, als mit der sie gefragt wurden usw.

Weiters können Geräte nicht nur unter verschiedenen Communities, sondern auch über mehrere IP-Adressen ansprechbar sein (falls sie mehrere davon besitzen). Geräte unterscheiden sich, ob sie nur unter einer IP-Adressen über SNMP erreichbar sind (welche dann die *Management-Adresse* genannt wird) oder über mehrere oder gar alle. Dazu kommt, dass abhängig von der IP-Adresse (und eventuell zusätzlich auch noch der Community), unter der ein Gerät angesprochen wird, wiederum unterschiedliche Daten geliefert werden können. Auch kann es vorkommen, dass die Antwort auf eine Anfrage von einer anderen IP-Adresse kommt, als an welche diese gesendet wurde. Auf all diese Aspekte muss im Discovery geachtet werden.

Ein weiterer zu berücksichtigender Punkt ist die Möglichkeit, dass manche Geräte so konfiguriert sein können, dass sie aus Sicherheitsgründen nur auf Anfragen antworten, die von vordefinierten IP-Adressen stammen. In solch einem Fall sollte dafür gesorgt werden, dass die Discovery-Maschine eine dieser IP-Adressen erhält bzw. die Geräte so eingestellt werden, dass sie auf die IP-Adresse der Discovery-Maschine reagieren.

Im Folgenden werden einige konkrete Informationen vorgestellt, die über SNMP bezogen werden können und die im Rahmen des Discoverys interessant sind.

### 3.3.1 SNMP Scan

Bevor Informationen von SNMP-Geräten abgefragt werden können, gilt es, diese erst einmal zu finden. Dafür kann beispielsweise eine Anfrage für die Variable `sysName` (`1.3.6.1.2.1.1.5.0`), welche den Namen des jeweiligen Geräts enthält und generell von jeder SNMP-fähigen Komponente unterstützt wird, an alle IP-Adressen in einem Adressbereich geschickt werden, wofür zusätzlich eine passende Community angegeben werden muss. Gibt es mehrere Communities, die in Frage kommen, so muss der ganze Vorgang für jede dieser wiederholt werden. Antwortet ein Gerät auf diese

Anfrage, so ist es als SNMP-Gerät (das auf die entsprechende Community reagiert) identifiziert. Da, wie im vorigen Abschnitt erwähnt, Geräte auch von anderen IP-Adressen bzw. mit anderen Communities antworten können, als zuvor in der Anfrage spezifiziert, darf das Empfangen einer Antwort alleine nicht als hinreichend angesehen werden, sondern diese Antwort muss zuerst dekodiert und analysiert werden.

SNMP-Anfragen können auch an Broadcast-Adressen gesendet werden, worauf alle SNMP-Geräte im entsprechenden Subnetz antworten sollten (so sie auch unter der gegebenen Community ansprechbar sind). Dies wird *Broadcast SNMP* genannt. Damit können theoretisch alle oder zumindest mehrere SNMP-Geräte in einem Subnetz durch das Senden eines einzelnen Pakets gefunden werden. In der Praxis antworten jedoch viele Geräte nicht auf Anfragen an Broadcast-Adressen bzw. werden Pakete an diese gar nicht erst ins Zielsubnetz weitergeleitet. Broadcast SNMP wird beispielsweise in [53] verwendet.

### 3.3.2 Geräteinformationen

Über SNMP können einige Basisinformationen (definiert in der `MIB-II`, siehe *RFC 1213* [52]) über jedes Gerät ausgelesen werden. Dazu gehören der Name eines Geräts (`sysName`, `1.3.6.1.2.1.1.5.0`), eine Beschreibung (`sysDescr`, `1.3.6.1.2.1.1.1.0`), dessen Standort (`sysLocation`, `1.3.6.1.2.1.1.6.0`) und Typ (`sysObjectID`, `1.3.6.1.2.1.1.2.0`), eine Kontaktperson (`sysContact`, `1.3.6.1.2.1.1.4.0`) sowie einiges mehr.

Diese Informationen können Hinweise zur Klassifizierung von Geräten geben. Die Variable `sysServices` (`1.3.6.1.2.1.1.7.0`) gibt an, auf welchen Layern ein Gerät operiert. Ist das erste Bit der Variable gesetzt, bedeutet dies, dass es auf Layer 1 arbeitet usw. Bei Switches muss das zweite Bit gesetzt sein und zusätzlich die `BRIDGE-MIB` (siehe *RFC 4188* [60]) unterstützt werden. Bei Routern muss das dritte Bit gesetzt sein und zusätzlich die Variable `ipForwarding` (`1.3.6.1.2.1.4.1.0`) den Wert `1` enthalten, um anzuzeigen, dass das Gerät Routing-Funktionen ausführt.

Weiters können Geräte durch die MIBs kategorisiert werden, die sie implementiert haben. Wird beispielsweise die `PRINTER-MIB` (siehe *RFC 3805* [7]) angeboten, so kann davon ausgegangen werden, dass es sich um einen Drucker handelt.

### 3.3.3 Interface Tables

Das `ifTable` (`1.3.6.1.2.1.2.2`), ebenfalls definiert in der `MIB-II`, gibt alle Interfaces eines Geräts an. Mit den Informationen aus dieser Tabelle lässt

sich feststellen, welche (über ihre MAC-Adresse) im Netzwerk gefundenen Interfaces zu jeweils einem Gerät zusammengehören.

Wichtige Spalten im `ifTable` sind der `ifIndex` (`1.3.6.1.2.1.2.2.1.1`) – dieser wird verwendet, um an anderen Stellen auf das entsprechende Interface zu verweisen –, die zugeordnete MAC-Adresse (`ifPhysAddress`, `1.3.6.1.2.1.2.2.1.6`), der Typ des Interfaces (`ifType`, `1.3.6.1.2.1.2.2.1.3`), die Geschwindigkeit (`ifSpeed`, `1.3.6.1.2.1.2.2.1.5`) und eine textuelle Beschreibung (`ifDescr`, `1.3.6.1.2.1.2.2.1.2`).

Prinzipiell sind vor allem Ethernet-Interfaces (`ethernetCsmacd`, `6`) interessant. Aber auch virtuelle Interfaces sind zu beachten, welche z.B. als Management Interfaces oder für virtuelle Routing Engines benutzt werden. Für diese wird von verschiedenen Geräten leider kein einheitlicher Typ verwendet wird, sondern beispielsweise `other` (`1`), `softwareLoopback` (`24`) oder `propVirtual` (`53`) – manchmal werden diese sogar als Ethernet-Interfaces bezeichnet. Virtuelle Interfaces für eine Link Aggregation Group haben den Typ `ieee8023adLag` (`161`).

Ein Gerät kann auch anhand seiner Interfaces klassifiziert werden. Hat jenes beispielsweise ein Funk-Interface (`ieee80211`, `71`), könnte es sich um einem WLAN Access Point handeln. Hat es z.B. ISDN (`isdn`, `63`), Frame Relay (`frameRelay`, `32`) oder ATM (`atm`, `73`) Interfaces, könnte es ein WAN-Router sein. Leider gibt es durch die große Anzahl unterschiedlicher Interface-Typen gewisse Unschärfen, was für eine bestimmte Art von Interface angegeben werden kann. So könnte beispielsweise ein Ethernet-Interface auch den Typ `fastEther` (`62`), `fastEtherFX` (`69`) oder `gigabitEthernet` (`117`) haben. Das heißt, manche Geräte verwenden den generischen Typ `ethernetCsmacd`, während andere den Typ spezifischer definieren. Dies muss beim Discovery berücksichtigt werden.

Zusätzliche Informationen zu jedem Interface können sich im `ifXTable` (`1.3.6.1.2.1.31.1.1`), definiert in der `IF-MIB` (siehe *RFC 2863* [51]), befinden (falls ein Gerät die genannte MIB unterstützt). Dazu gehört der Name des Interfaces (`ifName`, `1.3.6.1.2.1.31.1.1.1.1`), aus welchem sich noch am ehesten auf den pyhsischen Port eines Switches (z.B. `Fa3/2` oder `fe.3.2` für Port 2 von Slot 3 eines Switches) rückschließen lässt. Die Einträge im `ifXTable` und im `ifTable` werden über den `ifIndex` korreliert.

Weder das `ifTable`, noch das `ifXTable` besitzen Informationen über IP-Adressen. Diese Daten sind separat im `ipAddrTable` (`1.3.6.1.2.1.4.20`) gespeichert. Dessen wichtigsten Spalten sind die eigentliche IP-Adresse (`ipAdEntAddr`, `1.3.6.1.2.1.4.20.1.1`), die dazugehörige Subnetz-

maske (`ipAdEntNetMask`, `1.3.6.1.2.1.4.20.1.3`) und das Interface (`ipAdEntIfIndex`, `1.3.6.1.2.1.4.20.1.2`), dem die IP-Adresse zugeordnet ist. Um Informationen über die Interfaces eines Geräts zu gewinnen, sollten also zumindest das `ifTable` und das `ipAddrTable` ausgelesen und deren Daten korreliert werden.

Mit den Adressen aus dem `ipAddrTable` lässt sich auch auf vorhandene Subnetze schließen. Neue Subnetze können erkannt werden, wenn ein Gerät ein Interface in einem Subnetz besitzt, das noch unbekannt ist. Daher lässt sich beispielsweise über die Interfaces eines Routers bestimmen, welche Subnetze dieser kennt und verbindet – ob er zwischen diesen auch tatsächlich routet, muss extra mittels seiner Routing Tables festgestellt werden.

So wichtig die Informationen aus den Interface Tables auch sind, so bilden sie doch die (physische) Realität nicht in allen Fällen hundertprozentig ab. Nicht nur, dass die Tabellen auch virtuelle Interfaces enthalten können (die verschiedensten Zwecken dienen), sondern es sind diese manchmal auch nicht entsprechend gekennzeichnet und stattdessen als physische Interfaces angegeben. Umgekehrt kann aber auch der Fall eintreten, dass ein Gerät unter einer IP-Adresse erreichbar ist, für die es kein Interface gibt (bzw. die im `ipAddrTable` nicht aufscheint). Weiters muss beim Discovery beachtet werden, dass es laut den Interface Tables oft mehrere Interfaces mit der gleichen MAC-Adresse gibt. Im Netzwerk kann aber nur diese eine MAC-Adresse entdeckt, d.h. z.B. an einem Switch Port gehört werden. Daher ist zu überlegen, wie dies auf die dazugehörigen Interfaces umgelegt werden soll (z.B. welches von denen wird als mit dem Switch Port verbunden angesehen).

### 3.3.4 Connection Tables

Die Tabellen `tcpConnTable` (`1.3.6.1.2.1.6.13`) und `udpTable` (`1.3.6.1.2.1.7.5`) enthalten Informationen über momentan etablierte Verbindungen (bei TCP) bzw. über offene TCP und UDP Ports auf denen gelauscht wird. Daraus kann über die Port-Nummern geschlossen werden, welche Services ein Gerät anbietet.

### 3.3.5 Routing Tables

Das `ipRouteTable` (`1.3.6.1.2.1.4.21`) gibt an, welche Routen einem Gerät bekannt sind. Dabei wird zwischen *direkten* und *indirekten* Routen unterschieden. Im ersten Fall ist das Gerät direkt mit dem Zielsubnetz verbunden, im zweiten gibt der Next Hop den nächsten Router auf dem Weg zum Zielsubnetz an. Weitere Informationen (z.B. Metriken) sind ebenfalls verfügbar. Diese Informationen werden für den Aufbau der Layer 3 Topologie benötigt (z.B. welche Subnetze es gibt und welche Router dafür zuständig

sind) und sinnvollerweise nur von Routern ausgelesen. Es kann jedoch so auch bestimmt werden, ob etwa ein Endgerät andere Routen, neben dem Default Gateway, kennt.

Vorsicht ist geboten, die Subnetzmaske des Zielsubnetzes in einem Routing-Eintrag nicht mit der Subnetzmaske des Interfaces, über das Pakete an das Zielsubnetz gesendet werden, zu verwechseln, da sich diese bei indirekten Routen unterscheiden können. In [77] wird gezeigt, wie die Informationen aus den Routing Tables genau verarbeitet werden müssen, um die Pfade auf Layer 3 zu bestimmen.

### 3.3.6 Address Forwarding Tables

Wie in Abschnitt *2.5.2 Switching* beschrieben, werden die zum Switching notwendigen Informationen – über jeweils welchen Port eines Switches Frames an eine bestimmte MAC-Adresse gesendet werden sollen – in den AFTs gespeichert. Diese Daten bilden beim Discovery die Basis zur Bestimmung der Layer 2 Topologie. Über die AFTs können auch im Netz aktive Geräte gefunden werden (wobei aber von diesen nur deren MAC-Adresse festgestellt werden kann), die sonst mit anderen Methoden nicht auffindbar sind (weil z.B. keine direkte Kommunikation zwischen ihnen und der Discovery-Maschine möglich ist).

Um die AFTs auszulesen muss unterschiedlich vorgegangen werden, je nachdem ob VLANs auf einem Switch definiert sind oder nicht. Werden keine VLANs verwendet, so muss das in der `BRIDGE-MIB` definierte `dot1dTpFdbTable` (`1.3.6.1.2.1.17.4.3`) ausgelesen werden. In dieser Tabelle ist der Index jeder Zeile die gehörte MAC-Adresse, zu welcher Informationen gespeichert sind. Diese MAC-Adresse ist dabei in dezimaler Form und durch Punkte getrennt (entsprechend dem Format einer OID) gegeben und muss in die übliche hexadezimale Schreibweise konvertiert werden. In der Spalte `dot1dTpFdbPort` (`1.3.6.1.2.1.17.4.3.1.2`) ist die Nummer des Ports gespeichert, an dem die MAC-Adresse der entsprechenden Zeile gehört wurde. Die Spalte `dot1dTpFdbStatus` (`1.3.6.1.2.1.17.4.3.1.3`) gibt den Status des jeweiligen Eintrags an. Für das Discovery sind generell nur Zeilen mit dem Zustand `learned` (`3`) interessant. Ein anderer Wert wäre beispielsweise `self` (`4`), der angibt, dass der Eintrag die MAC-Adresse des jeweiligen Ports selbst beschreibt.

Zusätzlich muss beachtet werden, dass in allen Tabellen der `BRIDGE-MIB` die Ports eines Switches über ihre *Port Number* identifiziert werden, während es ansonsten üblich ist, den `ifIndex` eines jeden Interfaces zu verwenden (vgl. Abschnitt *3.3.3 Interface Tables*). Um diese Port-Nummern mit den entsprechenden Interface-Indizes zu korrelieren, muss

das `dot1dBasePortTable` (`1.3.6.1.2.1.17.1.4`) ausgelesen werden. Die Spalte `dot1dBasePort` (`1.3.6.1.2.1.17.1.4.1.1`) enthält dabei die Port-Nummer und `dot1dBasePortIfIndex` (`1.3.6.1.2.1.17.1.4.1.2`) den `ifIndex` des dazugehörigen Interfaces, wie er in anderen MIBs und Tabellen (z.B. dem `ifTable`) verwendet wird.

Werden auf einem Switch VLANs verwendet, so muss zusätzlich unterschieden werden, von welchem Hersteller das Gerät stammt. *Cisco*-Geräte verwenden das eingangs des Kapitels erwähnte Community Indexing. In diesem Fall muss das `dot1dTpFdbTable` mehrfach, und zwar für jedes VLAN separat, ausgelesen werden, wobei immer die ID des entsprechenden VLANs an den Community String anzuhängen ist – je nach aktuellem VLAN enthält die Tabelle andere Daten. So lässt sich jede gefundene MAC-Adresse nicht nur einem Port sondern auch einem VLAN zuordnen. Anzumerken ist, dass auch das `dot1dBasePortTable` ein Mal pro VLAN abgefragt werden muss, denn bei jedem Durchgang sind nur jene Port-Nummer-Interface-Index-Zuordnungen in der Tabelle enthalten, die Ports betreffen, welche dem entsprechenden VLAN zugeordnet sind. Natürlich müssen für diese Vorgehensweise die vorhandenen VLANs bereits bekannt sein (siehe Abschnitt *3.3.8 VLAN Tables*).

Die Standard-basierte Methode zum Auslesen von AFTs mit VLAN-Informationen ist die Verwendung der `Q-BRIDGE-MIB` (siehe *RFC 4363* [41]), welche die `BRIDGE-MIB` erweitert. In diesem Fall muss das `dot1qTpFdbTable` (`1.3.6.1.2.1.17.7.1.2.2`) anstelle des `dot1dTpFdbTable` ausgelesen werden. Die Spalten `dot1qTpFdbPort` (`1.3.6.1.2.1.17.7.1.2.2.1.2`) und `dot1qTpFdbStatus` (`1.3.6.1.2.1.17.7.1.2.2.1.3`) haben dabei die gleiche Funktion wie ihre Gegenstücke aus dem `dot1dTpFdbTable`. Der einzige Unterschied ist die doppelte Indizierung der Zeilen im `dot1qTpFdbTable`, zuerst mit der *FDB ID* (Kennung der *Forwarding Database*) und danach mit der dem Eintrag zugehörigen MAC-Adresse. Daher muss der Zeilen-Index zuerst in diese beiden Teile aufgespalten werden. Die MAC-Adresse kann wie bei der Vorgehensweise ohne VLANs verarbeitet werden, die FDB ID muss dem entsprechenden VLAN zugeordnet werden (pro VLAN wird intern eine eigene Datenbank verwendet). Dies wird noch näher in Abschnitt *3.3.8 VLAN Tables* beschrieben.

Normalerweise werden AFT-Einträge nach 5 Minuten gelöscht, wenn innerhalb dieser Zeit kein Frame mit der entsprechenden MAC-Adresse empfangen wurde. Daher ist es empfehlenswert, vor dem Auslesen der AFTs zuerst mit allen bekannten Geräten zu kommunizieren, um die Tabellen zu befüllen. Dies kann z.B. über das Senden von Pings (siehe Abschnitt *3.6.1 Ping*) oder ARP Requests (siehe Abschnitt *3.4.1 ARP Ping*) geschehen.

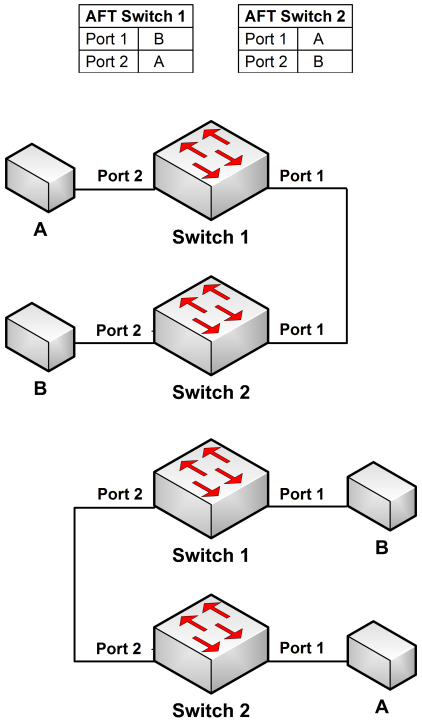


**Abbildung 3.1:** Zweideutige AFT-Einträge

Wie eingangs dieses Abschnitts erwähnt, können die Daten aus den AFTs zur Bestimmung der Layer 2 Topologie verwendet werden. Hierzu wurden in der Literatur einige Algorithmen vorgeschlagen, wie beispielsweise in [44], [10], [45], [53], [5], [78] oder [58]. Auf diese soll hier jedoch nicht näher eingegangen werden, sondern es sei auf die genannten Quellen verwiesen. Außerdem ist anzumerken, dass viele der vorgestellten Algorithmen nur mit gewissen Einschränkungen bzw. nur unter bestimmten Vorbedingungen funktionieren. *Abbildung 3.1* soll prinzipiell zeigen, dass in speziellen Fällen die Informationen aus den AFTs alleine nicht ausreichen, um die Topologie korrekt zu bestimmen. In Anbetracht der Anzahl der in einem durchschnittlichen LAN vorhandenen Geräte, ist der in der Grafik vorgestellte Fall in der Praxis jedoch äußerst unwahrscheinlich.

### 3.3.7 ARP Tables

Damit nicht immer neue ARP Requests ausgeführt werden müssen, werden einmal erfasste Informationen in den ARP Tables (standardmäßig für 30 Sekunden) gespeichert. Diese Daten können aus dem `ipNetToMediaTable` (`1.3.6.1.2.1.4.22`) ausgelesen werden, in welchem MAC-Adressen (`ipNetToMediaPhysAddress`, `1.3.6.1.2.1.4.22.1.2`) den

IP-Adressen (`ipNetToMediaNetAddress`, `1.3.6.1.2.1.4.22.1.3`) zugeordnet werden.

Da bei verschiedenen Discovery-Methoden entweder nur IP-Adressen oder nur MAC-Adressen (und selten beide gleichzeitig) gefunden werden, müssen diese korreliert werden. SNMP-fähige Geräte können die nötigen Daten über sich selbst direkt über das `ifTable` und das `ipAddrTable` anbieten. Alternativ dazu kann diese Zuordnung von der Discovery-Maschine durchgeführt werden, indem sie ARP Requests durchführt (siehe Abschnitt *3.4.1 ARP Ping*) – diese sind aber auf die lokale Broadcast Domain beschränkt. Die dritte Möglichkeit ist das hier beschriebene Auslesen von ARP Tables (vorzugsweise von Routern, da über diese jegliche Kommunikation in andere Subnetze geht). Der Vorteil dieser Methode besteht darin, dass damit auch MAC-Adressen von Geräten gefunden werden können, die nicht in der lokalen Broadcast Domain liegen (daher direkte ARP Requests nicht eingesetzt werden können), die nicht mit SNMP ansprechbar sind (deren MAC-Adressen daher nicht abgefragt werden können) oder mit denen aus sonstigen Gründen nicht direkt kommuniziert werden kann.

Problematisch im Umgang mit ARP Tables ist die eingangs erwähnte Kurzlebigkeit deren Einträge. Das heißt, wird länger als die Timeout-Zeitspanne nicht mit einem bestimmten Gerät kommuniziert, verfällt der entsprechende Eintrag. Weiters kann die Größe von ARP Tables beschränkt sein, sodass Einträge verworfen werden, wenn zu viele neue dazukommen. Aus diesem Grund ist es empfehlenswert, zuerst mit den Geräten zu kommunizieren, deren MAC-Adressen bestimmt werden sollen (bzw. maximal mit der Anzahl von Geräten, für deren Einträge genug Platz in einem ARP Table ist). Das kann etwa über das Senden von Pings (siehe Abschnitt *3.6.1 Ping*) geschehen, bevor die ARP Tables (z.B. von Routern) ausgelesen werden.

In Abschnitt *3.4.2 Virtual Device Detection* wird beschrieben, welche anderen Informationen aus MAC-Adressen an sich gezogen werden können.

### 3.3.8 VLAN Tables

Um die Layer 1/2 Topologie bestimmen zu können, müssen die in einem Netzwerk verwendeten VLANs erkannt werden. Zusätzlich werden die definierten VLANs auch als Eingabedaten für andere Discovery-Techniken (vgl. z.B. Abschnitt *3.4.1 ARP Ping*) benötigt.

Leider verwenden Switches unterschiedlicher Hersteller verschiedene Tabellen, um die VLAN-Informationen anzubieten. Herstellerunabhängig ist die `Q-BRIDGE-MIB`. In dessen `dot1qVlanCurrentTable` (`1.3.6.1.2.1.17.7.1.4.2`) werden alle auf einem Switch definierten VLANs aufgelistet, wobei die VLAN ID in der Spalte `dot1qVlanIndex`

(1.3.6.1.2.1.17.7.1.4.2.1.2) vorzufinden ist. Weiters wird jedem VLAN eine eigene *Forwarding Database* (*FDB*) zugeordnet, dessen Nummer in dot1qVlanFdbId (1.3.6.1.2.1.17.7.1.4.2.1.3) angegeben ist. Beim Auslesen der AFTs (siehe Abschnitt *3.3.6 Address Forwarding Tables*) können deren Einträge über diese FDB-Nummer mit dem jeweiligen VLAN korreliert werden. Beim Auslesen des dot1qVlanCurrentTable muss darauf geachtet werden, dass dieses den TimeFilter Mechanismus (siehe *RFC 4502* [84]) verwendet, d.h. der Zeilenindex einen Zeitstempel enthält, der es ermöglicht, nur Zeilen abzufragen, die sich seit einem bestimmten Zeitpunkt geändert haben.

Über welche Ports zu einem VLAN gehörige Frames ausgegeben werden dürfen (mit oder ohne VLAN Tag), geben die Spalten dot1qVlanCurrentEgressPorts (1.3.6.1.2.1.17.7.1.4.2.1.4) und dot1qVlanCurrentUntaggedPorts (1.3.6.1.2.1.17.7.1.4.2.1.5) an. Frames aus einem VLAN dürfen über alle Ports in dot1qVlanCurrentEgressPorts gesendet werden, ist ein Port zusätzlich in dot1qVlanCurrentUntaggedPorts, so wird das VLAN Tag vor dem Ausschicken entfernt, ansonsten wird das VLAN getaggte Frame unverändert ausgegeben. Die Information, welches VLAN das Default VLAN für jeden Port ist (d.h. auf einem Port empfangene, ungetaggte Frames bekommen dieses VLAN zugewiesen), kann der dot1qPortVlanTable (1.3.6.1.2.1.17.7.1.4.5) entnommen werden. Dass in diesen Tabellen die Nummern der Switch Ports verwendet werden und diese mittels der dot1dBasePortTable in den jeweiligen ifIndex übersetzt werden müssen, wie in Abschnitt *3.3.6 Address Forwarding Tables* beschrieben, ist zu beachten.

Wenn VLANs auf einem Switch statisch definiert wurden (was meist der Fall ist), so kann aus dem dot1qVlanStaticTable (1.3.6.1.2.1.17.7.1.4.3) auch der Name (dot1qVlanStaticName, 1.3.6.1.2.1.17.7.1.4.3.1.1) zu jedem VLAN bestimmt werden (die Tabelle ist über die VLAN ID indiziert). Beim Discovery muss beachtet werden, dass ein VLAN (mit der gleichen ID) auf verschiedenen Switches unterschiedliche Namen haben kann.

Viele Geräte der Firma *Cisco* verwenden hingegen nicht die Q-BRIDGE-MIB, sondern das proprietäre *VLAN Trunking Protocol* (*VTP*). Daher sind Informationen über definierte VLANs der CISCO-VTP-MIB (siehe [88]) und der CISCO-VLAN-MEMBERSHIP-MIB (siehe [43]) zu entnehmen. Die vorhandenen VLANs können aus dem vtpVlanTable (1.3.6.1.4.1.9.9.46.1.3.1) ausgelesen werden, wobei die ID eines jeden VLANs über den Index der jeweiligen Zeile bestimmt werden kann und dessen Name in der Spalte vtpVlanName (1.3.6.1.4.1.9.9.46.1.3.1.1.4) zu finden ist. Weiters sollten nur Ethernet-VLANs – die Spalte vtpVlanType

(1.3.6.1.4.1.9.9.46.1.3.1.1.3) hat den Wert ethernet (1) – berücksichtigt werden. Welches VLAN welchem Port zugeordnet ist, gibt das vmMembershipTable (1.3.6.1.4.1.9.9.68.1.2.2) an.

### 3.3.9 STP Tables

Um den Spanning Tree von Switches nachvollziehen zu können, müssen die entsprechenden Daten ausgelesen werden, welche auch Teil der BRIDGE-MIB sind. Diese Informationen können beim Bestimmen der Layer 1/2 Topologie behilflich sein, vor allem indem Uplink Ports (die Verbindungen zwischen Switches) erkannt werden. Weiters werden damit auch deaktivierte Verbindungen erfasst, für welche sonst (z.B. in den AFTs) keine Daten verfügbar sind.

Im STP werden alle Switches durch ihre *Bridge ID* identifiziert. Diese 8 Bytes lange Kennung ergibt sich aus der dot1dStpPriority (1.3.6.1.2.1.17.2.2.0), welche die ersten beiden Bytes ausmacht, und der dot1dBaseBridgeAddress (1.3.6.1.2.1.17.1.1.0), welche die restlichen 6 Bytes ausmacht. Wenn die Bridge ID eines Switches gleich dem Wert von dot1dStpDesignatedRoot (1.3.6.1.2.1.17.2.5) ist, so ist dieser Switch der Root Switch in einem Spanning Tree.

Die Informationen zu den einzelnen Ports eines Switches sind im dot1dStpPortTable (1.3.6.1.2.1.17.2.15) gespeichert. Ports werden über ihre 4 Byte lange *Port ID* identifiziert, welche analog zur Bridge ID aus der dot1dStpPortPriority (1.3.6.1.2.1.17.2.15.1.2) und der Port-Nummer in dot1dStpPort (1.3.6.1.2.1.17.2.15.1.1) gebildet wird. Diese Port-Nummer muss, wie in Abschnitt *3.3.6 Address Forwarding Tables* beschrieben, über das dot1dBasePortTable mit einem Interface korreliert werden. Weiters enthält das dot1dStpPortTable für jeden Port die Information, mit welchem Port (dot1dStpPortDesignatedPort, 1.3.6.1.2.1.17.2.15.1.9) welches Switches (dot1dStpPortDesignatedBridge, 1.3.6.1.2.1.17.2.15.1.8) dieser verbunden ist, genauso wie in Abschnitt *2.5.2 Switching* beschrieben. Beachtet werden muss, dass – wie ebenfalls im genannten Abschnitt bereits erläutert – Verbindungen im Spanning Tree immer nur auf einer Seite angegeben werden (der Baum also gerichtete Kanten hat).

In [76] wird detailliert beschrieben, wie die Spanning Tree Informationen auszulesen und zu verarbeiten sind und daraus in Kombination mit den Daten aus den AFTs die Layer 2 Topologie bestimmt werden kann.

### 3.3.10 Link Aggregation

Anhand des Typs eines Interfaces lässt sich zwar feststellen, ob es das virtuelle Interface für eine Link Aggregation Group ist, aber um die dazugehörigen physischen Interfaces (sowie deren Partner auf der anderen Seite der Link Aggregation) zu finden, welche gemeinsam die Link Aggregation Group bilden, müssen die Informationen aus der `IEEE8023-LAG-MIB` (siehe [33]) ausgelesen und analysiert werden. Dies ist nötig, um die Layer 1 Topologie vollständig aufbauen zu können. Für weitere Details dazu sei auf die angeführte Definition der MIB verwiesen.

### 3.3.11 Virtuelle Router

Um virtuelle Router erkennen und physischen Routern zuordnen zu können (zwecks vollständiger Abbildung der Layer 3 Topologie), können die Informationen aus der `VRRP-MIB` (siehe *RFC 2787* [36]) für VRRP bzw. der `CISCO-HSRP-MIB` (siehe [15]) für HSRP verwendet werden, wobei aber hier ebenfalls nur auf die genannten Quellen verwiesen sei.

Wenn nur die Adressen von virtuellen Routern erkannt werden sollen (und nicht auch die dahinterstehenden physischen Router), so kann auf das in Abschnitt *3.4.2 Virtual Device Detection* vorgestellte Verfahren zurückgegriffen werden.

## 3.4 Address Resolution Protocol (ARP)

ARP wurde bereits in Abschnitt *2.7.1 Address Resolution Protocol (ARP)* beschrieben. In diesem Abschnitt wird hingegen gezeigt, wie es über seine eigentliche Funktion – das Zuordnen von MAC-Adressen zu IP-Adressen – hinaus im Discovery eingesetzt werden kann.

### 3.4.1 ARP Ping

Die in dieser Arbeit als *ARP Ping* bezeichnete Methode steht für nichts weiter als das Senden von ARP Requests. Dabei wird für jede zu untersuchende IP-Adresse ein Broadcast Frame gesendet – falls darauf mit einem ARP Reply geantwortet wird, ist klar, dass sich hinter der gegebenen IP-Adresse ein aktives Gerät verbirgt. So kann ein IP-Adressbereich nach Geräten gescannt werden.

Die Vorteile dieser Methode liegen darin, dass ARP als Grundlage der TCP/IP-Kommunikation nicht nur sehr schnell ist, sondern dass Geräte einerseits auf ARP Requests auch antworten *müssen* (und diese nicht wie z.B. ICMP Pakete ignorieren können) und dass andererseits ARP Frames

gewöhnlich auch nicht von Firewalls blockiert werden (diese also mit dieser Technik umgangen werden können). Das ergibt sich aus der Tatsache, dass ohne funktionierendem ARP ein Datenaustausch generell unmöglich wäre. Dass nicht nur die Information geliefert wird, welche Geräte aktiv sind, sondern zusätzlich auch, welche MAC-Adresse der jeweiligen IP-Adresse zugeordnet ist, ist ein zusätzlicher Vorteil.

Als nachteilig muss man ansehen, dass die zu untersuchenden IP-Adressen bekannt sein müssen und dass nur Geräte in der selben Broadcast Domain erreichbar sind. Normalerweise wird ARP nur verwendet, wenn sich die gesuchte IP-Adresse im selben Subnetz wie der Fragesteller befindet. Daran muss man sich jedoch nicht halten und kann auch IP-Adressen aus anderen Subnetzen abfragen. Wenn sich die dazugehörigen Geräte in der selben Broadcast Domain befinden (also zumindest im selben VLAN sind), werden sie antworten. Dies gibt auch Hinweise für ein Mapping der Layer 3 auf die Layer 2 Topologie.

Dieses Verfahren kann jedoch auch erweitert werden, indem VLAN getaggte ARP Request Frames gesendet werden. Hierfür muss einerseits der Switch Port, an dem die Discovery Maschine angeschlossen ist, dies auch erlauben bzw. entsprechend konfiguriert sein und andererseits müssen die zu untersuchenden VLANs bekannt sein (siehe Abschnitt *3.3.8 VLAN Tables*). Sind diese Voraussetzungen jedoch erfüllt, können Geräte aus allen VLANs, die sich in der selben „physischen“ Broadcast Domain befinden, gefunden werden und die Einschränkung auf die „virtuelle“ Broadcast Domain des VLANs, in dem sich die Discovery-Maschine befindet, fällt weg. Mit dieser Technik kann sehr schnell ein komplettes (auf Layer 1/2 zusammenhängendes) LAN gescannt werden, ohne dass die gesendeten Frames blockiert oder ignoriert werden könnten. Zusätzlich erhält man auch die Information, welches Gerät sich in welchem VLAN befindet.

Ein Problem beim Einsatz von ARP Ping können ARP Proxies darstellen, da durch deren Verhalten inkorrekte MAC-IP-Zuordnungen gefunden werden. Eine einfache Heuristik, um ARP Proxies zu erkennen, wäre etwa, Geräte mit MAC-Adressen, denen sehr viele, mehr als eine bestimmte Anzahl (wobei es aber keinen allgemeingültigen Wert dafür gibt) an IP-Adressen zugeordnet sind, als ARP Proxies zu markieren.

Da das Senden von ARP Frames Datenverkehr am Netzwerk mit den jeweiligen Geräten erzeugt, kann das ARP Ping auch verwendet werden, um AFTs zu befüllen. Das heißt, mit allen gefundenen Geräten im Netzwerk wird kommuniziert, wodurch die Switches, die zwischen der Discovery-Maschine und dem jeweiligen Gerät liegen, dessen MAC-Adresse hören und in ihre AFTs aufnehmen.

### 3.4.2 Virtual Device Detection

Dieser Abschnitt beschreibt, wie mit Hilfe der von ARP gelieferten Informationen – den MAC-Adressen – virtuelle Geräte identifiziert werden können. Hierbei handelt es sich nicht um eine Technik, die auf der Funktionsweise des ARP-Protokolls aufbaut, sondern um eine Heuristik, die auf die darüber beschafften Daten (die MAC-Adressen) angewandt wird. Daher kann diese Methode generell mit allen MAC-Adressen verwendet werden, egal von welcher Quelle sie stammen.

In [73] wird beschrieben, wie virtuelle Maschinen des Typs *VMware* erkannt werden können, indem der Vendor ID Teil einer MAC-Adresse (die ersten drei Bytes) analysiert wird. Eine VMware liegt vor, wenn die Vendor ID `00:05:69`, `00:0C:29` oder `00:50:56` beträgt, da dies die Herstellerkennungen sind, die der Firma VMware zugeordnet sind. Es handelt sich dabei aber nur um eine Heuristik, da MAC-Adressen per Software geändert werden könnten und daher der hier vorgestellte Sachverhalt nicht zwingend zutreffen muss. Weiters könnte diese Technik auch der Zuordnung von VM Hosts und Guests dienen, wenn an einem Switch Port mehrere virtuelle MAC-Adressen (Guests) und eine „normale“ MAC-Adresse (Host) gefunden werden.

Analog lässt sich dieses Vorgehen auch zur Erkennung virtueller Router einsetzen. Die MAC-Adresse eines virtuellen Routers muss bei VRRP mit `00:00:5E:00:01` (siehe *RFC 3768* [27]) und bei HSRP mit `00:00:0C:07:AC` (siehe *RFC 2281* [42]) beginnen. Daraus kann geschlossen werden, dass es sich bei allen passenden MAC-Adressen sowie den dazugehörigen IP-Adressen um virtuelle Router handelt. Um diese auch physischen Routern zuordnen zu können, werden weitere Informationen benötigt (siehe Abschnitt *3.3.11 Virtuelle Router*).

## 3.5 Internet Protocol (IP)

IP – dessen Funktionsweise in Abschnitt *2.6.2 Routing* behandelt wurde – ist die Grundlage für viele weitere Protokolle, wie z.B. ICMP, TCP und UDP. Während IP nur zum Übertragen der Daten dieser Protokolle verwendet wird, bietet es dennoch einige Optionen, die für das Discovery nützlich sein können. Das in diesem Abschnitt Vorgestellte lässt sich daher im Zusammenhang mit allen Protokollen einsetzen, die auf IP basieren.

Da IP auf Layer 3 operiert und geroutet wird, können alle darauf aufbauenden Techniken (z.B. SNMP, ICMP Ping, TCP Scans, UDP Scans etc.) auch von einem entfernten Standort, etwa über das Internet, verwendet werden. Dennoch wird davon abgeraten und empfohlen, die Discovery-Maschine direkt im zu untersuchenden LAN zu platzieren, da Netzwerke meist nach

außen hin durch Firewalls abgesichert sind, welche oft Pakete blockieren, die zum Discovery benötigt werden.

### 3.5.1 Source Routing

Wie bereits in Abschnitt *2.6.2 Routing* beschrieben, kann durch Source Routing der Pfad eines Pakets vorgegeben und dadurch das Netzwerk eventuell aus verschiedenen „Perspektiven“ betrachtet werden. Source Routing ist vor allem beim Discovery im Internet nützlich, da in LANs die Routen meist nicht so komplex sind, als dass mit Source Routing viele neue Informationen gewonnen werden könnten. In [23] wird Source Routing beispielsweise dazu eingesetzt, um im Internet Backup-Routen zu finden, die gewöhnlicherweise von den Routern nicht verwendet werden.

### 3.5.2 OS Fingerprinting

Das sogenannte *OS Fingerprinting* bezeichnet Heuristiken zum Erkennen des Betriebssystems auf einer entfernten Maschine. OS Fingerprinting verwendet dabei zwar keine Daten von IP selbst, sondern von darauf aufbauenden Protokollen (z.B. ICMP oder TCP).

OS Fingerprinting funktioniert, indem bestimmte Aspekte von Paketen (z.B. Flags, Sequenznummern, Codes etc.) – je nachdem um welches Protokoll es sich konkret handelt – analysiert werden. Die Spezifikation der jeweiligen Protokolle definiert zwar, wie die einzelnen Felder in Paketen zu verwenden sind, wobei aber nicht alles bis ins letzte Detail vorgegeben ist. Daher können verschiedene Implementierungen eines Protokolls (in verschiedenen Betriebssystemen) sich leicht abweichend verhalten (z.B. wie Sequenznummern generiert werden). Da die Verhaltensweisen einzelner Betriebssysteme bekannt sind, können durch die Beobachtung von Paketen Rückschlüsse auf das möglicherweise verwendete Betriebssystem gezogen werden. Dies wird beispielsweise in [2], [85] und [57] beschrieben.

## 3.6 Internet Control Message Protocol (ICMP)

ICMP wird im TCP/IP-Stack zu Diagnose- und Steuerungszwecken verwendet und wurde bereits im Abschnitt *2.7.2 Internet Control Message Protocol (ICMP)* beschrieben. ICMP ist außerdem eines der im Discovery am häufigsten verwendeten Protokolle, da dessen Einsatz einfach ist und meist unterstützt wird. Dennoch werden auch ICMP-Pakete aus Sicherheitsgründen manchmal von Firewalls blockiert bzw. Geräte so konfiguriert, auf entsprechende Anfragen nicht zu antworten (siehe [2] und [47]). In diesem Abschnitt werden einige auf ICMP basierende Techniken vorgestellt.

### 3.6.1 Ping

Eines der bekanntesten Tools zur Netzwerkadministration ist `Ping` (siehe *RFC 2151* [38]). Eigentlich gedacht zum Testen von Netzwerkverbindungen, kann dieses bzw. die darunterliegende Methode im Discovery zum Finden von aktiven Geräten verwendet werden. Dabei wird eine ICMP `Echo Request` Nachricht an eine IP-Adresse geschickt. Verbirgt sich hinter dieser ein aktiver Host, so sollte dieser mit einem `Echo Reply` antworten (wobei dieser zusätzlich genau jene Daten enthält, die zuvor mit der Anfrage an das Gerät mitgesendet wurden). So kann ein ganzer IP-Adressbereich durchprobiert werden. Von jeder IP-Adresse, von der eine Antwort kommt, kann angenommen werden, dass diese zu einem aktiven Gerät gehört.

Diese Vorgehensweise ist nicht nur sehr einfach, sondern die ICMP Echo Funktion ist auch weit verbreitet (d.h. sehr viele Geräte antworten auf ein Ping), da sie häufig innerhalb eines Netzwerkes zu diversen Diagnosezwecken eingesetzt wird.

Da Ping Datenverkehr mit den Zielgeräten erzeugt, kann diese Technik auch eingesetzt werden, um beispielsweise ARP Tables oder AFTs mit Informationen zu befüllen (vgl. Abschnitte *3.3.6 Address Forwarding Tables* und *3.3.7 ARP Tables*). Antwortet ein Gerät auf ein Ping, so sendet es diese Antwort zumindest über den Switch Port, an dem es angeschlossen ist, wodurch die MAC-Adresse des Geräts in die AFT des Switches eingetragen wird. Damit zuvor die `Echo Request` Nachricht das Gerät überhaupt erst erreichen kann, muss der Router, der im selben Subnetz wie das Zielgerät liegt (vorausgesetzt dieses und die Discovery-Maschine liegen in verschiedenen Subnetzen und die Kommunikation geht daher über einen oder mehrere Router), einen ARP Request durchführen, um die MAC-Adresse des Zielgeräts zu ermitteln und das Paket dann an diese direkt weiterschicken zu können. Dies bewirkt aber zusätzlich, dass die IP- und MAC-Adresse des Zielgeräts in das ARP Table des Routers eingetragen werden.

Bei Pings kann auch mit „gefälschten“ Absenderadressen gearbeitet werden. Bei der zuvor beschriebenen Art ARP Tables zu befüllen, läuft die Kommunikation nur über den Router, der auf der Route zwischen Discovery-Maschine und Zielgerät liegt. Sollen aber auch ARP Tables von anderen Routern gefüllt werden (falls es mehrere Router im selben Subnetz wie das Zielgerät gibt), so kann ein ICMP `Echo Request` Paket von der Discovery-Maschine an den gewünschten Router geschickt werden, das jedoch als Absender nicht die Discovery-Maschine, sondern das zu untersuchende Zielgerät eingetragen hat. Für den Router sieht es so aus, als ob die Anfrage von diesem Gerät käme und er antwortet auch entsprechend diesem. Dafür muss er aber einen ARP Request ausführen, was zum Befüllen seiner ARP Table führt.

### 3.6.2 Broadcast Ping

Pings können nicht nur an „normale“ IP-Adressen, sondern auch an Broadcast-Adressen gesendet werden. In diesem Fall wird dann von einem *Broadcast Ping* gesprochen. Dabei stehen zur Auswahl: die Limited Broadcast Address `255.255.255.255` (mit der aber nur das lokale Subnetz erreicht werden kann), die jeweilige Network Address (z.B. `192.168.0.0/24`) oder die Directed Broadcast Address (z.B. `192.168.0.255/24`) eines jeden Subnetzes.

Idealerweise sollten auf ein Paket an eine Broadcast-Adresse alle Geräte im entsprechenden Subnetz antworten, wodurch alle darin aktiven Geräte über das Senden von nur einer Nachricht identifiziert werden können und nicht erst alle IP-Adressen durchprobiert werden müssen. Es hängt jedoch stark von den einzelnen Geräten (bzw. deren Typ, Betriebssystem etc.) ab, auf welche der drei genannten Adressen sie reagieren bzw. ob sie überhaupt auf Broadcast-Pakete antworten. Außerdem werden gerade Broadcast-Pakete meist von Firewalls blockiert bzw. von Routern gar nicht erst ins Zielsubnetz weitergeleitet. Dadurch ist diese theoretisch einfache und effektive Methode im praktischen Einsatz eher unzuverlässig. Verwendet wird diese Technik beispielsweise in [86] und [72].

Broadcast Pings können auch zum sogenannten *Subnet Guessing* eingesetzt werden, wie es in [72] beschrieben wird. Dabei wird eine Heuristik verwendet, die Broadcast-Adressen von möglichen Subnetzen durchprobiert. Falls auf eine Anfrage mehr als eine Antwort kommt, so wird geschlossen, dass es sich um eine gültige Broadcast-Adresse handelte und somit das dazugehörige Subnetz existieren muss.

### 3.6.3 Mask und Timestamp

Neben dem zuvor genannten ICMP `Echo Request/Reply` gibt es noch einige andere ICMP-Nachrichten, die sich zum Discovery eigenen. Bei einem `Mask Request` wird beispielsweise nach der Subnetzmaske des Zielgeräts gefragt, ein `Timestamp Request` fordert einen Zeitstempel an. Kommt auf diese Anfragen an eine IP-Adresse eine Antwort, so wurde ein aktives Gerät identifiziert. Bei einem `Mask Request/Reply` wird als zusätzliche Information die Subnetzmaske eines Geräts mitgeliefert, was bei der Bestimmung von Subnetzen bzw. der Layer 3 Topologie behilflich sein kann.

Der Vorteil der gerade genannten ICMP-Nachrichten ist deren seltene Verwendung, wodurch sie eventuell nicht von Firewalls blockiert werden. Weil diese Nachrichten aber kaum benutzt werden, werden sie andererseits auch nur von wenigen Geräten unterstützt – viele Computer beantworten der-

artige Anfragen also gar nicht. Diese Nachrichten werden beispielsweise in [86], [2] und [47] beschrieben.

### 3.6.4 Traceroute

Ein weiteres, oft verwendetes Tool ist `Traceroute` (siehe *RFC 2151* [38]). Der dahinterstehende Algorithmus sendet dabei mehrere Pakete an eine IP-Adresse. Meist werden dazu ICMP-Pakete (von *Microsoft Windows*-basierten Tools) verwendet, es können aber auch UDP- (unter *Unix* und *Linux*) oder seltener TCP-Pakete sein. Dabei wird zuerst ein Paket mit einer TTL von 1 verschickt. Dies führt dazu, dass die Lebenszeit des Pakets beim ersten Router am Weg zum Ziel abläuft, das Paket verworfen und eine ICMP `Time Exceeded` an die Discovery-Maschine zurückgesandt wird. Dadurch, dass diese ein Paket vom Router erhält, kennt sie nun den ersten Router auf dem Weg zum Zielgerät. Als nächstes wird ein Paket mit einer TTL von 2 geschickt, wodurch sich der zweite Router zu erkennen gibt. Dies wird so lange fortgeführt, bis das Zielgerät selbst antwortet. Am Ende dieses Vorganges sind alle Router auf der Strecke zwischen der Discovery-Maschine und dem Zielgerät bekannt.

Mit Traceroute kann die Layer 3 Topologie ganz oder teilweise bestimmt werden, ohne dass auf SNMP-Daten von den Routern selbst zurückgegriffen werden muss. Im Zusammenhang mit Traceroute tritt das Problem auf, dass mehrere IP-Adressen jeweils eines Routers als separate Geräte erkannt werden. Auf eine Methode zur Lösung dieses Problems wird gesondert in Abschnitt *3.8.3 IP Alias Resolution* eingegangen. Bei Traceroute bietet sich auch der Einsatz von Source Routing (siehe Abschnitt *3.5.1 Source Routing*) an, um verschiedene Routen zwischen Geräten zu entdecken. Traceroute wird beispielsweise in [86] und [72] verwendet.

## 3.7 Transmission Control Protocol (TCP)

Die Funktionsweise von TCP wurde bereits in Abschnitt *2.6.3 Datenübertragung* betrachtet. Nun soll dessen möglicher Einsatz im Discovery beleuchtet werden. Hierbei wird TCP zum Finden von aktiven Geräten verwendet, indem versucht wird diese zum Antworten auf TCP-Pakete zu bewegen. Durch die Analyse der Antworten können weitere Informationen gewonnen werden. TCP-basierte Methoden können benutzt werden, falls andere (z.B. auf ARP oder ICMP aufbauende) nicht die gewünschten Ergebnisse liefern können (deren Pakete beispielsweise von Firewalls blockiert werden etc.).

Bei TCP-Scans werden nicht nur einzelne IP-Adressen untersucht, sondern es müssen auch die Ports angegeben werden, die von der jeweiligen IP-Adresse abgefragt werden sollen. Daher sind mehrere Durchgänge (einer

pro Port) notwendig. Antwortet ein Gerät, so kann meist (abhängig von der konkret eingesetzten Scan-Methode) zusätzlich zur Erkennung des aktiven Geräts auch festgestellt werden, welche Ports offen und geschlossen sind, was Rückschlüsse auf angebotene Services zulässt. Gelingt es überhaupt, eine vollständige TCP-Verbindung aufzubauen, senden viele Services eine initiale Nachricht (genannt *Hello Message*). Diese kann analysiert werden, was weitere Details (z.B. Server-Software und dessen Version) über den angebotenen Dienst offenbaren kann.

TCP-Scans sind besonders effektiv, wenn ein Gerät bestimmte Services anbietet, da in diesem Fall auf entsprechende TCP-Pakete geantwortet werden muss (weil die Dienste sonst nicht funktionieren würden). Bietet ein Gerät unter einer gewissen Port-Nummer kein Service an, dann ist es meist so konfiguriert, nicht zu antworten, bzw. es werden in solch einem Fall Pakete oft von Firewalls blockiert. Zusätzlich werden manchmal generell (d.h. auch bei aktiven Services) nur Pakete akzeptiert bzw. durchgelassen, die von gewissen vordefinierten Absenderadressen stammen.

Verschiedene TCP-Scans werden in diesem Abschnitt vorgestellt. Für eine detailliertere Behandlung sei beispielsweise auf [2], [47] und [48] verwiesen.

### 3.7.1 TCP SYN Scan

Beim *TCP SYN Scan* wird ein TCP-Paket mit gesetztem `SYN` Flag an eine IP-Adresse und einen Port geschickt. Idealerweise sollte ein Port gewählt werden, von dem angenommen wird, dass sich dahinter ein aktives Service verbirgt. Meist wird es erforderlich sein, mehrere Port-Nummern für jede IP-Adresse durchzuprobieren, also in mehreren Durchgängen zu scannen.

Das Senden eines Pakets mit gesetztem `SYN` Flag ist nichts anderes als der erster Schritt im Three-Way Handshake (siehe *2.6.3 Datenübertragung*) zum Aufbau einer TCP-Verbindung. Ist der Port, an den das Paket gesandt wurde, geschlossen, so sollte der Computer (so sich hinter der benutzten IP-Adresse ein aktives Gerät verbirgt) mit einem TCP-Paket antworten, in welchem das `RST` Flag gesetzt ist um anzuzeigen, dass unter diesem Port keine Verbindung hergestellt werden kann. Ist der Port jedoch offen, so antwortet das Gerät mit einen Paket, in dem sowohl das `SYN` als auch das `ACK` Flag gesetzt sind – das ist gemäß dem Standard der nächste Schritt im Verbindungsaufbau. In beiden Fällen erhält die Discovery-Maschine eine Antwort und das Gerät kann als aktiv identifiziert werden. Zusätzlich kann je nach Art der Antwort erkannt werden, ob der entsprechende Port offen oder geschlossen ist.

Offene Ports lassen darauf schließen, dass das Gerät gewisse Services anbietet. Daher kann in so einem Fall die Verbindung komplett hergestellt werden,

indem auch noch der dritte Schritt des Verbindungsaufbaus ausgeführt wird. Dieser veranlasst dann, wie zuvor erwähnt, in manchen Fällen den Dienst zum Senden einer Hello Message, welche analysiert werden kann. Diese hängt jedoch vom konkreten Service ab und muss daher für jedes Protokoll separat betrachtet werden. Natürlich kann über eine hergestellte Verbindung nach den Regeln des dahinterstehenden Protokolls weiter kommuniziert werden, um so an zusätzliche Informationen zu gelangen. Aber auch hier muss je nach Protokoll unterschiedlich vorgegangen werden.

Aus Sicherheitsgründen sind die meisten Geräte derart konfiguriert, dann nicht zu antworten, wenn eine Anfrage an einen geschlossenen Port gesendet wurde, um ihre Existenz nicht preiszugeben, indem sie kein Paket mit gesetztem `RST` Flag senden. Analog dazu blockieren Firewalls oft TCP-Pakete, die an Hosts gerichtet sind, von denen sie wissen, dass sie unter der entsprechenden Port-Nummer keinen Dienst anbieten bzw. anbieten dürfen.

Die eben beschriebene Methode eignet sich daher besonders gut zum Erkennen von Servern, die eine Servicefunktion ausüben, indem sie entsprechende Dienste anbieten.

### 3.7.2 TCP ACK Scan

Eine weitere Methode ist der sogenannte *TCP ACK Scan*, bei dem ein TCP-Paket mit gesetztem `ACK` Flag an einen Port und eine IP-Adresse gesendet wird. So ein Paket bestätigt normalerweise den Erhalt von zuvor empfangenen Paketen. Da aber gar keine Verbindung vorliegt, kommt der Vorgang für das gefragte Gerät unerwartet. Es sollte daher mit einem Paket mit gesetzten `RST` Flag antworten (egal ob der Port dahinter offen oder geschlossen ist). Wird daher von einem Gerät eine Antwort erhalten, so lässt sich darauf schließen, dass es aktiv ist.

Der Vorteil dieser Vorgehensweise liegt darin, dass sie eventuell von manchen Firewalls nicht behindert wird, falls diese nur den Versuch eines Verbindungsaufbaus (also Pakete mit gesetztem `SYN` Flag) blockieren, aber nicht generell alle TCP-Pakete verwerfen. Nachteilig wirkt sich aus, dass Geräte oft – ebenfalls auch Sicherheitsgründen – so konfiguriert sind, gar nicht zu antworten, anstatt ein `RST`-Paket zu senden. Weiters kann mit dieser Methode nicht festgestellt werden, ob ein bestimmter Port offen oder geschlossen ist. Daher können auch keine Informationen über vorhandene Services gesammelt werden, sondern nur aktive Geräte identifiziert werden. Auch diese Methode funktioniert am besten, wenn ein Host unter dem gefragten Port ein Service anbietet.

### 3.7.3 Weitere Scans

Neben den beiden genannten TCP-Scans gibt es noch eine Reihe von weiteren, bei welchen verschiedene Varianten von gesetzten Flags bzw. anderen Optionen verwendet werden. Das Ziel ist jeweils, eine Kombination zu finden, die von Firewalls nicht gefiltert wird und auf die Hosts antworten (und sich so zu erkennen geben). Ansonsten ist das Vorgehen analog zu dem bereits beschriebenen. Daher sei an dieser Stelle für eine vertiefende Behandlung auf weitere Quellen verwiesen, wie beispielsweise [2].

### 3.7.4 TCP Echo

Eine Alternative zu den genannten Scans ist die Verwendung des *TCP Echo* Protokolls (siehe *RFC 862* [66]). Dieses funktioniert analog zum ICMP Echo (siehe Abschnitt *3.6.1 Ping*). Wird über Port 7 eine Verbindung aufgebaut, so werden alle darüber gesendeten Daten in der empfangenen Form wieder zurückgeschickt, bis die Verbindung abgebrochen wird. Ein derartiger Datenaustausch gibt die Existenz des Zielgeräts preis. Leider ist (aus Sicht des Discoverys) dieses Service jedoch nur selten auf Geräten aktiviert bzw. blockieren Firewalls dessen Pakete.

## 3.8 User Datagram Protocol (UDP)

Die Funktionsweise von UDP wurde ebenfalls bereits in Abschnitt *2.6.3 Datenübertragung* betrachtet. Im Discovery dient es auch hauptsächlich – analog zu TCP – zum Finden von Geräten. Einige UDP-basierte Techniken werden in diesem Abschnitt behandelt.

### 3.8.1 UDP Ping

Beim *UDP Ping* wird ein UDP-Paket an einen Port einer IP-Adresse gesandt. Ist dieser Port geschlossen (und befindet sich hinter der IP-Adresse ein aktives Gerät), so sollte eine ICMP `Port Unreachable` Nachricht zurückgeschickt werden. Der Erhalt derselben bestätigt die Existenz des Geräts. Es ist anzumerken, dass dies nicht für andere ICMP-Antwort-Nachrichten gilt (z.B. `Destination Unreachable` geschickt von einem Router). Diese deuten eher darauf hin, dass der Host *nicht* aktiv ist.

Wird das Paket an einen offenen Port gesendet, ist wahrscheinlich, dass es nicht dem Format des dahinterstehenden Services entspricht, daher vom Zielgerät einfach verworfen und gar keine Antwort zurückgeschickt wird. Daher lässt sich einerseits vom Fehlen einer Rückmeldung nicht auf inaktive IP-Adressen schließen, andererseits ist es bei dieser Technik wünschenswert, Ports zu testen, von denen angenommen wird, dass sich dahinter *kein* Dienst

verbirgt. Auch hier ist es meist empfehlenswert, den Scan in mehreren Durchgängen mit verschiedenen Port-Nummern durchzuführen.

Natürlich können auch Pakete an einen bestimmten Port geschickt werden, die den Regeln des dazugehörigen Protokolls folgen, wodurch eine Antwort darauf wahrscheinlicher wird. Diese Antwort lässt einerseits den Schluss zu, dass das entsprechende Service auf dem Zielgerät aktiv ist und andererseits lassen sich durch deren Analyse auch weitere Informationen über den Dienst (z.B. Server-Software samt Version) bestimmen. Ebenso kann die Kommunikation über das passende Protokoll weitergeführt werden, um zusätzliche Daten zu gewinnen. Dies alles fällt jedoch nicht mehr wirklich in den Bereich eines einfachen Scans, muss doch hier je nach Protokoll separat vorgegangen werden.

Der Vorteil von UDP-Scans liegt darin, dass sie nicht nur eine Alternative zu anderen (z.B. ARP- oder ICMP-basierten) Techniken darstellen, sondern auch in manchen Fällen Firewalls passieren können, wenn diese nur TCP-Pakete blockieren. Ebenso unterdrücken einige Geräte nur TCP- nicht aber ICMP-Antworten. Nachteile dieser Vorgehensweise sind wie gewöhnlich, dass einerseits auch UDP-Pakete von Firewalls blockiert werden bzw. Geräte so konfiguriert sein können, nicht mit einer ICMP `Port Unreachable` Nachricht zu antworten. Andererseits ist es schwieriger, Antworten von offenen Ports (und dementsprechend Informationen über Services) zu erhalten.

UDP-Scans werden im Detail in [2], [47] und [48] behandelt.

### 3.8.2 UDP Echo

Das *UDP Echo* Protokoll (siehe *RFC 862* [66]) funktioniert analog zum TCP Echo (siehe Abschnitt *3.7.4 TCP Echo*), mit dem Unterschied, dass keine Verbindung aufgebaut werden muss und alle an Port 7 geschickten UDP-Pakete umgehend 1:1 wieder retourniert werden (wodurch die Existenz des Zielgeräts bestätigt wird). Dies stellt ebenfalls eine Alternative zu anderen Scan-Methoden dar, ist aber genauso mit dem Nachteil behaftet, dass dieses Service kaum aktiviert ist bzw. Firewalls meist dessen Pakete blockieren.

### 3.8.3 IP Alias Resolution

Eine besondere Anwendung von UDP liegt in der sogenannten *IP Alias Resolution* (siehe [74]). Werden beispielsweise mittels Traceroute (siehe Abschnitt *3.6.4 Traceroute*) Router bestimmt, so wird vorerst jede Router-IP-Adresse als separater Router erkannt, obwohl es sich bei manchen IP-Adressen nur um verschiedene Interfaces eines einzigen Geräts handelt, sogenannte IP-Aliasse ein und desselben Routers. Um zu erkennen, welche IP-Adressen zu einem Gerät zusammengehören, kann die hier beschriebene Technik einge-

setzt werden, welche eine Möglichkeit zum Gruppieren von Interfaces darstellt, ohne dass auf SNMP-Daten von den Geräten selbst zurückgegriffen werden muss.

Um Aliasse zu erkennen, wird an jede Kandidaten-IP-Adresse ein UDP-Paket an einen Port gesendet, von dem angenommen wird, dass er geschlossen ist. Dies sollte eine ICMP `Port Unreachable` Nachricht als Antwort auslösen, wobei laut Spezifikation (siehe *RFC 1122* [9]) als Absenderadresse dieses ICMP-Pakets die des Interfaces eingetragen sein sollte, das auf der direkten Route zurück zur Discovery-Maschine liegt. Werden daher Pakete an verschiedene IP-Adressen geschickt, wobei die Antworten darauf hingegen jeweils von der gleichen IP-Adresse kommen, so können die entsprechenden IP-Adressen als zu einem Gerät gehörig betrachtet werden. Allerdings ist das beschriebene Verhalten nur ein empfohlenes, d.h. nicht verpflichtend. Daher kann diese Technik nur als Heuristik angesehen werden.

Die hier beschriebene Methode wird in [23] und [1] verwendet. In [74] werden zusätzlich noch einige weitere Techniken zur IP Alias Resolution vorgestellt.

## 3.9 Domain Name System (DNS)

Im Abschnitt *2.7.3 Domain Name System (DNS)* wurde bereits die Funktionsweise von DNS erklärt. Nun soll auf auf dessen Einsatz im Discovery eingegangen werden.

DNS-Abfragen müssen immer an einen DNS-Server gerichtet werden. Normalerweise ist in der Konfiguration eines Computers ein Default Name Server festgelegt und alle DNS-Operationen verwenden implizit diesen. Aber es können mittels spezieller DNS-Bibliotheken auch explizit andere oder mehrere DNS-Server abgefragt werden (sofern diese bekannt sind oder z.B. während des Discoverys identifiziert wurden).

### 3.9.1 Lookup und Reverse Lookup

Die Hauptfunktionen des DNS-Systems, nämlich das Suchen von IP-Adressen zu Netzwerknamen (Lookup) sowie umgekehrt das von Netzwerknamen zu IP-Adressen (Reverse Lookup), können im Discovery genau in dieser Form eingesetzt werden: um IP-Adressen zu Namen zu finden und vice versa (je nachdem welche Information gegeben ist, irgendwelche Eingabedaten als Ausgangsbasis sind auf jeden Fall nötig).

Weiters können DNS-Einträge Hinweise auf möglicherweise aktive Hosts geben. Wenn beispielsweise eine bestimmte IP-Adresse eingetragen ist, so ist doch wahrscheinlich, dass es dazu auch ein passendes Gerät im Netzwerk gibt

bzw. zumindest einmal gab, auch wenn es von der Discovery-Maschine nicht erreicht werden kann bzw. dieser nicht antwortet. Anzumerken ist außerdem, dass es auf Grund der Funktionsweise von DNS mit einem Reverse Lookup nicht möglich ist, alle Alias-Namen zu einer gegebenen IP-Adresse zu bestimmen, wie in Abschnitt *2.7.3 Domain Name System (DNS)* erklärt.

### 3.9.2 Zone Transfer

Eine weitere Möglichkeit, um an DNS-Informationen im Discovery zu gelangen, ist die Ausführung eines Zone Transfers von einem DNS-Server. Dies wird beispielsweise in [86] und [72] verwendet.

Zone Transfers haben den Vorteil, dass einerseits keine Eingabedaten (IP-Adressen oder Namen, die aufgelöst werden sollen) notwendig sind und andererseits auf einmal alle Daten geliefert werden, die der DNS-Server besitzt. Weiters können diese Daten analysiert und so z.B. alle Alias-Namen zu jeder IP-Adresse extrahiert werden.

Da Zone Transfers eigentlich zum Datenaustausch nur zwischen DNS-Servern gedacht sind, verweigern diese gewöhnlich aus Sicherheitsgründen diesbezügliche Anfragen, sofern sie nicht von einem anderen bekannten DNS-Server stammen. Daher kann die Discovery-Maschine meist nicht an diese Informationen gelangen, außer sie besitzt die nötigen Berechtigungen bzw. die DNS-Server wurden entsprechend konfiguriert, um der Discovery-Maschine den Zugriff zu erlauben.

### 3.9.3 Weitere Records

Im DNS werden auch eine Reihe anderer Einträge (z.B. `MX` oder `SRV`, siehe [79]) verwaltet, die zusätzliche Informationen über eine Domain (z.B. Mail-Server oder andere Services) enthalten, welche im Discovery entsprechend verwendet werden können (wobei diese Daten einzeln abgefragt oder über einen Zone Transfer bestimmt werden müssen).

## 3.10 Weitere Techniken

Um die Thematik möglichst umfassend zu behandeln, werden in diesem Abschnitt einige weitere mögliche Techniken für das Discovery beschrieben, wobei aber kein Anspruch auf Vollständigkeit erhoben wird. Es sollen Ideen geliefert werden, von welchen Quellen andere, möglicherweise für das Discovery nützliche Daten bezogen werden könnten. Für nähere Informationen zur Entwicklung konkreter Methoden, die diese Quellen ausnutzen, sei auf die angeführte Literatur bzw. Dokumentation oder Spezifikation verwiesen. In dem in dieser Arbeit vorgestellten Algorithmus (siehe Kapitel *4*

*Der Discovery-Algorithmus*) wird keine der in diesem Abschnitt genannten Techniken bzw. Datenquellen verwendet.

### 3.10.1 Discovery-Protokolle

Einige Protokolle, wie etwa das Hersteller-unabhängige *Link Layer Discovery Protocol* (*LLDP*, siehe [29]) oder das proprietäre *Cisco Discovery Protocol* (*CDP*, siehe [14]), wurden speziell dazu entwickelt, um die Layer 2 Topologie eines Netzwerkes zu erkunden. Diese werden vor allem auf Switches eingesetzt.

Diese Protokolle können ausgenutzt werden, indem entweder passiv nach dazugehörigen Paketen gelauscht wird sowie deren Informationen ausgewertet werden, oder indem die von diesen Protokollen gesammelten Daten via SNMP ausgelesen werden. Der Vorteil ist jedenfalls, dass mittels Discovery-Protokollen direkt die Layer 2 Topologie bestimmt werden kann und keine komplizierten Algorithmen zur deren Inferenz aus anderen Daten benötigt werden. Nachteilig ist jedoch, dass nicht damit gerechnet werden kann, dass diese Protokolle auch zwangsläufig auf einem bestimmten LAN eingesetzt werden.

### 3.10.2 Routing-Protokolle

Wie in Abschnitt *2.6.2 Routing* beschrieben, tauschen Router untereinander Daten unter Zuhilfenahme von Routing-Protokollen aus, von denen es eine Vielzahl gibt: z.B. *Open Shortest Path First* (*OSPF*), *Interior Gateway Routing Protocol* (*IGRP*), *Border Gateway Protocol* (*BGP*), *Exterior Gateway Protocol* (EGP) oder *Routing Information Protocol* (*RIP*).

Diese Protokolle liefern Informationen über die Layer 3 Topologie. Diesbezügliche Pakete können entweder passiv abgehört oder schon über sie gesammelte Informationen über SNMP ausgelesen werden, wie in Abschnitt *3.3.5 Routing Tables* beschrieben. Der Vorteil von passivem Lauschen liegt darin, dass auch Daten von Routern verarbeitet werden können, auf die über SNMP nicht zugegriffen werden kann. Allerdings ist das Auslesen über SNMP wesentlich einfacher und die Daten liegen in einer einheitlichen Form vor (egal über welches Protokoll sie bestimmt wurden), während beim passiven Abhören für jedes Protokoll unterschiedlich vorgegangen werden muss. In [86] wird u.a. eine Technik beschrieben, die nach RIP-Paketen lauscht.

### 3.10.3 Universal Plug and Play

*Universal Plug and Play* (*UPnP*, siehe [81]) definiert eine Reihe von Datenformaten und Protokollen, um verschiedenste Geräte in einem Netzwerk nach einem einheitlichen Schema ansteuern zu können. Die Bezeichnung

„plug and play“ im Namen soll andeuten, dass Geräte einfach ans Netzwerk angeschlossen werden können, danach sofort erkannt werden und anschließend verfügbar sind.

Eine Komponente von UPnP ist das Discovery von UPnP-Geräten, welches auf dem *Simple Service Discovery Protocol* (*SSDP*, siehe [11]) basiert. Dieses könnte beim Network Discovery verwendet werden, um UPnP-Geräte zu finden, wobei keinerlei Eingabedaten dafür notwendig sind (da es über Multicasts funktioniert).

UPnP ist jedoch hauptsächlich für Heimumgebungen konzipiert worden (z.B. zur Ansteuerung von Audio- oder Videogeräten im Heimnetzwerk) und wird in Firmenumgebungen eher nicht anzutreffen sein.

### 3.10.4 NetBIOS over TCP/IP

*NetBIOS over TCP/IP* (*NBT*, siehe *RFC 1001* [59]) definiert, wie die *NetBIOS*-Dienste über TCP/IP-Netzwerke verwendet werden können. Über das *NetBIOS Naming Service* (*NBNS*) können Informationen über Domains, NetBIOS Namen und Benutzer extrahiert werden. Auf NBT bauen auch weitere Services auf, wie etwa *Server Message Block* (*SMB*) bzw. *Common Internet File System* (*CIFS*).

Im Discovery können NBT-Pakete (bzw. Pakete darauf aufbauender Dienste) abgehört und analysiert oder auch NBNS direkt verwendet werden (etwa über Systemaufrufe oder das Programm `nbtstat`).

Der Nachteil von NBT ist, dass es nur in *Microsoft Windows*-Umgebungen eingesetzt wird und nur relativ wenige für das Discovery nützliche Informationen daraus gewonnen werden können. Dafür kann NBNS einerseits ohne Eingabeparameter verwendet werden (da es über Broadcasts funktioniert), andererseits kommen auf NBT basierende Dienste in *Microsoft Windows*-Netzwerken häufig vor. Weiters ist interessant, dass NBNS auch Daten über Benutzer verwaltet.

### 3.10.5 Web Based Enterprise Management

*Web Based Enterprise Management* (*WBEM*) definiert eine Reihe von Technologien und Standards zur Fernwartung von Geräten in einem Netzwerk (ähnlich wie SNMP). Die Implementierung und Erweiterung von WBEM für *Microsoft Windows*-Umgebungen heißt *Windows Management Instrumentation* (*WMI*, siehe [50]).

Über WMI können nahezu alle Konfigurationseinstellungen eines Computers sowohl lokal als auch über das Netzwerk ausgelesen werden. Liest man z.B.

die Daten eines Domain Controllers aus, so können u.a. Informationen über alle Hosts in dieser Domain gewonnen werden. Der Vorteil der Verwendung von WMI im Discovery ist das immense Ausmaß an Informationen, die damit über eine einheitliche Schnittstelle gewonnen werden können. Nachteile sind: WMI ist nur für *Microsoft Windows*-Umgebungen verfügbar (bzw. andere WBEM-Implementierungen nur für die jeweilige Plattform), die zu untersuchenden Geräte müssen bekannt und die entsprechenden Zugriffsrechte (etwa über einen Administrator-Account) vorhanden sein.

### 3.10.6 Konfigurationsfiles

Eine andere Möglichkeit, Informationen von einem Gerät zu gewinnen, ist Konfigurationsdateien (z.B. `/etc/network/interfaces` unter *Linux*) direkt von diesem über einen Netzwerk-Dateizugriff auszulesen. Dafür müssen die entsprechenden Verzeichnisse jedoch auch über das Netzwerk zugänglich sein (auf *Microsoft Windows* gibt es dazu etwa die standardmäßig vorhandenen, versteckten administrativen Shares `C$` und `ADMIN$`). Außerdem muss man passende Zugriffsrechte (etwa Username und Passwort eines berechtigten Benutzers) haben. Zusätzlich müssen die zu untersuchenden Geräte bekannt sein. Die mit dieser Vorgehensweise von Geräten extrahierten Konfigurationsinformationen können jedoch nicht einheitlich abgehandelt werden, d.h. mit jeder Art von Konfigurationsdatei muss separat umgegangen werden.

### 3.10.7 Remote Shells

Eine weitere Möglichkeit, Daten von einem Gerät zu abzufragen, ist ein Login dort manuell (bzw. durch das Discovery-Programm) über das Netzwerk auf der betreffenden Maschine – etwa mittels *Telnet*, *Remote Login* (*rlogin*), *Remote Shell* (*rsh*) oder *Secure Shell* (*SSH*) – und dann direkt Systemkommandos auszuführen (z.B. `ifconfig` bzw. `ipconfig`), welche die gewünschten Informationen liefern. Für dieses Vorgehen müssen die zu untersuchenden Geräte bekannt sowie passende Zugriffsrechte (etwa Username und Passwort eines berechtigten Benutzers) verfügbar sein. Vorteile sind dabei, dass über Systemkommandos (z.B. in einer *IOS* Shell von *Cisco* Switches) Informationen abgefragt werden können, die mit anderen Methoden (z.B. über SNMP) nur schwierig oder gar nicht erfragt werden können. Nachteilig hingegen ist die Vielzahl von verschiedenen Shells (z.B. *Microsoft Windows*, *Unix*, *Cisco IOS* etc.) mit jeweils eigener Syntax, die alle separat behandelt werden müssen.

### 3.10.8 Systemaufrufe

Neben dem Auslesen von Konfigurationsdateien und dem Zugriff auf Geräte über Remote Shells können Informationen auch über das Ausführen von

Systemaufrufen mittels der Programmierschnittstelle des Betriebssystems gewonnen werden. Dieses Vorgehen ist jedoch sehr spezifisch, weil abhängig von der verwendeten Plattform und den verfügbaren Systemaufrufen.

Ein Beispiel dafür ist der *Microsoft Windows*-Systemaufruf `NetWkstaUserEnum` (siehe [54]). Wird dieser (auf der Discovery-Maschine) ausgeführt, lassen sich damit die auf einer *anderen* Maschine gerade eingeloggten Benutzer feststellen. Damit dies funktioniert, muss der Aufruf im Kontext eines Benutzers stattfinden, der ausreichende Rechte dafür besitzt (etwa ein Domain-Administrator). Selbstverständlich müssen die zu untersuchenden Geräte bekannt sein. Interessant ist daran, dass mit dieser Vorgehensweise auch an Informationen gelangt werden kann (wie eben eingeloggte Benutzer), die mit „klassischen“ Methoden (z.B. Ping, Traceroute, Port Scans, Auslesen von Daten von Routern und Switches usw.) nicht verfügbar sind.

## 3.11 Sonderfälle

Beim Discovery gibt es Spezialfälle bzw. Problembereiche, die gelegentlich auftreten und mit denen umgegangen werden können muss. Einige davon wurden bereits in den vorigen Abschnitten erwähnt, hier werden diese nochmals zusammengefasst dargestellt, wobei jedoch kein Anspruch auf Vollständigkeit erhoben wird. Es wird versucht anzugeben, wie sich einzelne Fälle konkret im Discovery äußern und wie eventuell mit ihnen umgegangen werden könnte. Ziel dieses Abschnitts ist jedoch, vor allem darauf hinzuweisen, worauf prinzipiell geachtet werden muss, und nicht eine fertige Lösung für jeden Punkt zu präsentieren (sofern das nicht schon zuvor behandelt wurde). Das heißt, es werden an dieser Stelle unter Umständen Fragen aufgeworfen, diese aber nicht beantwortet – da deren Beantwortung von der individuellen Präferenz eines jeden Entwicklers abhängt.

Bei der Entwicklung eines Discovery-Systems ist es weniger wichtig, jeden möglichen Fall auch korrekt abbilden zu können (z.B. mehrere Interfaces eines Geräts auch als solche zu erkennen, anstatt als separate Geräte), sondern es kommt darauf an, sich möglichst aller Fälle bewusst zu sein (z.B. dass es Geräte mit mehreren Interfaces geben kann). Dann kann entschieden werden, ob lieber ein einfacherer Algorithmus, der dafür die Wirklichkeit nicht in allen Fällen korrekt abbildet, oder ein komplexerer Algorithmus mit mehr Genauigkeit entwickelt werden soll. Es muss klargestellt sein, wie ein Algorithmus mit bestimmten Fällen umgeht und mit welchen Ausgabedaten dann gerechnet werden kann. Wie genau diese dann sein sollen, liegt an den individuellen Anforderungen. Die können durchaus besagen, dass gewisse Sonderfälle nicht erkannt werden müssen, sofern sich der Algorithmus bei

deren Auftreten wie vorgesehen verhält nicht etwa „abstürzt“ oder komplett falsche Ergebnisse liefert.

Folgende Liste beschreibt (in keiner bestimmten Reihenfolge) einige der wichtigsten Sonderfälle:

- *Firewalls:* Firewalls stellen die größte Behinderung beim Discovery dar, da sie den Datenverkehr teilweise oder ganz blockieren können. Meistens werden sie zur Abschottung eines Netzwerkes nach außen eingesetzt, sodass das Discovery ungehindert möglich sein sollte, solange sich die Discovery-Maschine auf der selben Seite der Firewall befindet wie alle anderen Geräte (siehe [87]). Sollten dennoch an anderen Stellen Firewalls vorhanden sein, so kann entweder auf ein verteiltes Discovery (mit Discovery-Maschinen jeweils hinter den Firewalls) oder auf diverse Tricks zu deren Umgehung zurückgegriffen werden (wobei jedoch nicht garantiert werden kann, dass diese auch funktionieren). Zu diesen Tricks gehören etwa die in den Abschnitten *3.4.1 ARP Ping* (falls auf Layer 2 nicht gefiltert wird), *3.7.2 TCP ACK Scan* (falls nur Verbindungsaufbauversuche blockiert werden) und *3.8.1 UDP Ping* (falls nur TCP gefiltert wird) vorgestellten Techniken.

- *WLAN Access Points:* Wie WLAN Access Points im Discovery erkannt werden, hängt stark von deren Typ ab und ob sie SNMP unterstützen, sich darüber als Switch und/oder Router zu erkennen geben usw. Eine Möglichkeit, um WLAN Access Points heuristisch zu erkennen, bietet sich z.B., falls über SNMP ein Switch gefunden wird, der ein Funk-Interface (mit dem Typ `ieee80211`) besitzt. Hierbei muss bedacht werden, dass mit einem Funk-Interface mehrere Geräte gleichzeitig verbunden sein können (ohne dass ein Hub benötigt wird, wie bei normalen Interfaces). Werden WLAN Access Points nicht gesondert berücksichtigt, sollten sie als Switches bzw. Hubs (falls sie nicht SNMP-fähig sind) aufscheinen.

- *Multi-Homed Devices:* Sehr häufig kommt es vor, dass Geräte (vor allem Server) mehrere Interfaces haben, die dann separat (über ihre MAC-Adresse) im Netzwerk gefunden werden und es keine hundertprozentige Möglichkeit gibt, zusammengehörige Interfaces auch dem jeweils richtigen Gerät zuzuordnen. Falls Informationen über die Interfaces (z.B. über SNMP, aber auch WMI oder andere Methoden) zur Verfügung gestellt werden, lassen sich Interfaces entsprechend zusammenfassen. Sind solche Informationen nicht verfügbar, so wird es schwierig – komplizierte Heuristiken wären notwendig, um passende Hinweise zu erhalten. Können Interfaces ihren multi-homed Geräten nicht korrekt zugeordnet werden, so wird jedes Interface als separates Gerät erkannt.

- *Redundanter Anschluss:* Dies ist ein Spezialfall von multi-homed Geräten, die für zwei oder mehr Interfaces die gleiche MAC-Adresse verwenden, es also so aussieht, als ob ein Interface an mehreren Stellen gleichzeitig angeschlossen wäre. Durch die Funktionsweise der AFTs kann eine MAC-Adresse aber immer nur einem Port eines bestimmten Switches zugeordnet sein, daher kann es mehrere Discovery-Durchgänge benötigen, um diese Situation zu erkennen, falls es sich um Ports desselben Switches handelt. Hingegen wird die gleiche MAC-Adresse auf verschiedenen Switches immer gleichzeitig gefunden. Theoretisch könnte es sich dabei aber auch um zwei verschiedene Geräte handeln, die inkorrekt konfiguriert sind und deswegen die gleiche MAC-Adresse benutzen. Es obliegt dem konkreten Algorithmus, darüber zu entscheiden, welcher Fall als wahrscheinlicher erachtet und wie dementsprechend das Ergebnis interpretiert wird.

- *Link Aggregation:* Ähnlich zu den beiden vorigen Punkten ist der Einsatz von Link Aggregation, d.h. die Bündelung von mehreren physischen Interfaces. Darüber können Informationen vorhanden sein (z.B. über SNMP, aber auch über andere Quellen), die eine Zuordnung und Gruppierung der Interfaces auf einer oder beiden Seiten der Link Aggregation zulassen. In einem Algorithmus muss vorgesehen werden, wie reagiert werden soll, falls diese Informationen nicht verfügbar sind.

- *Doppelte MAC-Adressen*: Weiters muss festgelegt werden, wie ein Algorithmus sich verhalten soll, wenn ein Gerät (etwa über SNMP) angibt, mehrere Interfaces mit der gleichen MAC-Adresse zu besitzen, und diese dann beispielsweise an einer oder mehreren Stellen gefunden wird. Dies steht auch in Bezug zu den drei vorigen Punkten.

- *Cluster:* Im Server-Bereich werden oft Cluster eingesetzt. Im Idealfall würde ein Discovery-Algorithmus jede Maschine eines Clusters einzeln erkennen und diese den anderen zuordnen können. Da es verschiedenste Techniken gibt, um Cluster zu implementieren, kann hier keine bestimmte Vorgehensweise dazu vorgeschlagen werden.

- *Virtuelle Maschinen:* Auf Grund ihrer Flexibilität werden heutzutage (vor allem bei Servern) gerne virtuelle Maschinen eingesetzt (siehe [67]). Werden diese nicht als solche identifiziert bzw. Hosts und Guests einander nicht zugeordnet, so werden alle als eigenständige Geräte erkannt. In Abschnitt *3.4.2 Virtual Device Detection* wurde eine Heuristik zur Erkennung von virtuellen Maschinen des am häufigsten verwendeten Typs *VMware* (siehe ebenfalls [67]) vorgestellt. Zur korrekten Zuordnung von Guests und Hosts müssten weitere Heuristiken entwickelt werden.

- *Virtuelle Router:* Um die Ausfallssicherheit zu erhöhen werden oft virtuelle Router (VRRP/HSRP) eingesetzt. Dazugehörige Adressen können mittels der in Abschnitt *3.4.2 Virtual Device Detection* beschriebenen Methode entdeckt werden. Sollen virtuelle Router auch physischen Routern zugeordnet werden, müssen zusätzlich entsprechende Informationen z.B. über SNMP bestimmt werden (siehe Abschnitt *3.3.11 Virtuelle Router*). Werden virtuelle Router nicht korrekt identifiziert und zugeordnet, kann es zu Fehlinterpretationen bei der Analyse von Routen kommen.

- *Unrouted Subnets:* In einem Netzwerk können für spezielle Zwecke auch ungeroutete Subnetze vorkommen, falls die Geräte darin nur untereinander (und nicht in andere Netze) kommunizieren müssen. Geräte in diesen Subnetzen können prinzipiell gefunden werden, wenn sich die Discovery-Maschine in der selben Broadcast Domain befindet. Ein Problem ist es aber, diese Subnetze überhaupt erst zu erkennen (falls sie nicht explizit schon als Eingabedaten angegeben werden), da über sie in keinem Router Informationen darüber vorhanden sind. Gleiches gilt auch für Subnetze, für die Router zuständig sind, auf deren Informationen kein Zugriff besteht. Solche Subnetze könnten etwa über passives Abhören des Datenverkehrs oder mittels Subnet Guessing (siehe Abschnitt *3.6.2 Broadcast Ping*) gefunden werden.

- *Network Address Translation:* Befinden sich Geräte hinter einem NAT-Router, so gibt es praktisch keine Möglichkeit, direkt nach diesen zu suchen. Einzig mittels Heuristiken (siehe z.B. [6]) können einige Hinweise erlangt werden. Komplett umgangen werden kann dieses Problem, falls eine Discovery-Maschine (auch) hinter dem NAT-Router platziert werden kann.

- *Fehlerfälle:* Generell sollte ein Algorithmus auch mit Fehlerfällen (etwa doppelt vergebenen IP-Adressen, falschen Subnetzmasken etc.) umgehen können. Idealerweise werden diese als solche erkannt und gemeldet, zumindest sollten sie aber zu keinen falschen Ergebnissen führen.

## 3.12 Leitfaden

In diesem Abschnitt soll zusammenfassend eine Übersicht der wichtigsten der hier vorgestellten Methoden, inklusive deren Eigenschaften sowie Vor- und Nachteilen, geliefert werden. Die verschiedenen Techniken werden sowohl nach dem Protokoll, auf dem sie basieren (*3.12.1 Übersicht nach Protokoll*), als auch nach der Art der Informationen, die sie liefern (*3.12.2 Übersicht nach Art der Information*), angeführt. Diese Aufstellung soll den Vergleich der Methoden erleichtern sowie einen Leitfaden zur deren Auswahl und Zusammenstellung für eine bestimmte Aufgabe bieten.

### 3.12.1 Übersicht nach Protokoll

SNMP-basierte Methoden:

- **SNMP Scan**
  - *Eingabe*: Zu untersuchende IP-Adressen und SNMP Communities
  - *Ausgabe*: Aktive SNMP-Geräte inkl. der jeweils passenden SNMP Community
  - *Vorteile*: Findet SNMP-Geräte
  - *Nachteile*: Ein Durchgang pro SNMP Community erforderlich
- **Broadcast SNMP Scan**
  - *Eingabe*: Zu untersuchende Subnetze und SNMP Communities
  - *Ausgabe*: Aktive SNMP-Geräte inkl. der jeweils passenden SNMP Community
  - *Vorteile*: Findet alle SNMP-Geräte in einem Subnetz über das Senden von nur einem Paket
  - *Nachteile*: Geräte antworten u.U. nicht auf Broadcast-Pakete, Broadcast-Pakete werden u.U. gar nicht erst ins Zielsubnetz weitergeleitet
- **Geräteinformationen**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Basisinformationen (Name, Beschreibung, Typ etc.) zum Gerät
  - *Vorteile*: Von allen SNMP-Geräten unterstützt, kann bei der Klassifizierung von Geräten helfen
  - *Nachteile*: Liefert keine für das weitere Discovery besonders nützlichen Informationen
- **Interface Tables**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Informationen über die Interfaces des Geräts
  - *Vorteile*: Wird benötigt um einzelne Interfaces Geräten zuzuordnen, kann zum Erkennen neuer Subnetze verwendet werden (über die IP-Adressen von Interfaces)

  - *Nachteile*: Informationen reichen nicht aus, um tatsächlich vorhandene physische Interfaces zu enumerieren

- **Connection Tables**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Offene Verbindungen und (TCP und UDP) Ports auf denen gelauscht wird
  - *Vorteile*: Liefert Informationen über vorhandene Services
  - *Nachteile*: Keine

- **Routing Tables**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Bekannte Subnetze und (direkte und indirekte) Routen zu diesen
  - *Vorteile*: Liefert Informationen über die Layer 3 Topologie
  - *Nachteile*: Keine

- **Address Forwarding Tables**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Zuordnung von MAC-Adressen zu Switch Ports (und ev. VLANs)
  - *Vorteile*: Wird benötigt zum Erkennen der Layer 2 Topologie, kann zum Finden von Geräten verwendet werden (wobei nur MAC-Adressen gefunden werden), kann zur Zuordnung von Geräten zu VLANs verwendet werden
  - *Nachteile*: Informationen in den AFTs sind meist nicht hundertprozentig vollständig; je nachdem, ob und wie VLANs unterstützt werden, müssen verschiedene Techniken zum Auslesen verwendet werden (keine einheitliche Vorgehensweise für alle Fälle möglich)

- **ARP Tables**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Zuordnung von MAC-Adressen zu IP-Adressen

  - *Vorteile*: Erlaubt die Zuordnung von MAC-Adressen zu IP-Adressen, kann zum Finden von Geräten verwendet werden
  - *Nachteile*: Einträge in ARP Tables sind sehr kurzlebig (standardmäßiges Timeout von 30 Sekunden), ARP Tables können eine begrenzte Größe haben

- **VLAN Tables**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Auf dem Gerät definierte VLANs
  - *Vorteile*: Wird benötigt um die vorhandenen VLANs zu erkennen, liefert Informationen für die Layer 2 Topologie
  - *Nachteile*: Je nach Hersteller und Typ des Switches müssen verschiedene Techniken zum Auslesen verwendet werden (keine einheitliche Vorgehensweise für alle Fälle möglich)

- **STP Tables**
  - *Eingabe*: IP-Adresse des zu untersuchenden Geräts, passende SNMP Community
  - *Ausgabe*: Benachbarte Switches im Spanning Tree und Informationen darüber, welche Ports miteinander verbunden sind
  - *Vorteile*: Wird benötigt um den Spanning Tree aufzubauen, liefert Informationen für die Layer 2 Topologie
  - *Nachteile*: Probleme, falls Lücken im ermittelten Spanning Tree bestehen, weil beispielsweise auf gewisse Switches kein Zugriff besteht oder diese STP nicht unterstützen (z.B. manche WLAN Access Points)

ARP-basierte Methoden:

- **ARP Ping**
  - *Eingabe*: Zu untersuchende IP-Adressen
  - *Ausgabe*: Aktive Geräte, IP-MAC-Zuordnungen
  - *Vorteile*: Schnell; Geräte müssen antworten bzw. Frames dürfen nicht blockiert werden; nicht von Firewalls betroffen, außer diese filtern auch auf Layer 2; liefert zusätzlich die MAC-Adresse zu jeder IP-Adresse
  - *Nachteile*: Funktioniert nur im lokalen VLAN, funktioniert nicht über Router bzw. das Internet

- **ARP Ping + VLAN**
  - *Eingabe*: Zu untersuchende IP-Adressen und VLANs
  - *Ausgabe*: Aktive Geräte, IP-MAC-VLAN-Zuordnungen
  - *Vorteile*: Wie *ARP Ping*, kann alle VLANs im lokalen LAN durchsuchen, liefert zusätzlich das VLAN zu jedem Gerät
  - *Nachteile*: VLANs müssen bekannt sein, Switch Port muss entsprechend konfiguriert sein, ein Durchgang pro VLAN, funktioniert nicht über Router bzw. das Internet

ICMP-basierte Methoden:

- **Ping**
  - *Eingabe*: Zu untersuchende IP-Adressen
  - *Ausgabe*: Aktive Geräte
  - *Vorteile*: Standardmethode zum Testen von Verbindungen, meist zu Diagnose-Zwecken aktiviert (besonders bei Infrastrukturkomponenten)
  - *Nachteile*: Manchmal Geräte konfiguriert nicht zu antworten, oft von Firewalls blockiert
- **Directed Broadcast Ping**
  - *Eingabe*: Zu untersuchende Subnetze
  - *Ausgabe*: Aktive Geräte in den jeweiligen Subnetzen
  - *Vorteile*: Findet alle Geräte in einem Subnetz über das Senden von nur einem Paket, kann zum Subnet Guessing benutzt werden
  - *Nachteile*: Geräte oft konfiguriert nicht zu antworten, meist von Firewalls blockiert, Broadcast-Pakete werden u.U. gar nicht erst ins Zielsubnetz weitergeleitet
- **Limited Broadcast Ping**
  - *Eingabe*: Die Limited Broadcast Adresse
  - *Ausgabe*: Aktive Geräte im selben Subnetz
  - *Vorteile*: Keine Eingabedaten notwendig
  - *Nachteile*: Kann nur Geräte im selben Subnetz finden, Geräte oft konfiguriert nicht zu antworten

- **Mask**
  - *Eingabe*: Zu untersuchende IP-Adressen
  - *Ausgabe*: Aktive Geräte, Subnetzmasken zu den IP-Adressen
  - *Vorteile*: Liefert zusätzlich die Subnetzmasken
  - *Nachteile*: Geräte meist konfiguriert nicht zu antworten, manchmal von Firewalls blockiert
- **Traceroute**
  - *Eingabe*: Zu untersuchende IP-Adressen
  - *Ausgabe*: Aktive Geräte, Router auf dem Weg zu den Geräten
  - *Vorteile*: Liefert Daten zur Layer 3 Topologie
  - *Nachteile*: Langsam; liefert keine Information, welche Router-Interfaces zusammengehören

TCP-basierte Methoden:

- **TCP SYN Scan**
  - *Eingabe*: Zu untersuchende IP-Adressen und Ports
  - *Ausgabe*: Aktive Geräte, offene Ports, ev. Service-Informationen
  - *Vorteile*: Liefert auch Service-Informationen (offene Ports bzw. Analyse von Hello Messages, die allerdings je nach Service unterschiedlich sind); funktioniert gut bei Geräten, die Services anbieten; Alternative, falls *ICMP Ping* nicht anwendbar ist
  - *Nachteile*: Oft Geräte konfiguriert nicht zu antworten bzw. Pakete von Firewalls gefiltert, ein Durchgang für jeden Port notwendig
- **TCP ACK Scan**
  - *Eingabe*: Zu untersuchende IP-Adressen und Ports
  - *Ausgabe*: Aktive Geräte
  - *Vorteile*: Wird von Firewalls, die nur Verbindungsaufbauversuche blockieren, nicht gefiltert; Alternative, falls Geräte nicht mit ICMP Ping erreichbar sind
  - *Nachteile*: Oft Geräte konfiguriert nicht zu antworten bzw. Pakete von Firewalls gefiltert, ein Durchgang für jeden Port notwendig
- **TCP Echo**
  - *Eingabe*: Zu untersuchende IP-Adressen

  - *Ausgabe*: Aktive Geräte
  - *Vorteile*: Einfach, alternative Möglichkeit zu anderen Scans
  - *Nachteile*: Echo-Service auf Geräten meist nicht aktiviert bzw. von Firewalls blockiert

UDP-basierte Methoden:

- **UDP Scan**
  - *Eingabe*: Zu untersuchende IP-Adressen und Ports
  - *Ausgabe*: Aktive Geräte, ev. Service-Informationen
  - *Vorteile*: Kann Firewalls passieren, die nur TCP filtern; kann Service-Informationen liefern, falls ein Service antwortet (jedoch Anfrage im richtigen Format für das jeweilige Protokoll erforderlich); Alternative, falls *ICMP Ping* nicht anwendbar ist
  - *Nachteile*: Oft Geräte konfiguriert nicht zu antworten bzw. Pakete von Firewalls gefiltert, ein Durchgang für jeden Port notwendig, langsamer als TCP-Scans
- **UDP Echo**
  - *Eingabe*: Zu untersuchende IP-Adressen
  - *Ausgabe*: Aktive Geräte
  - *Vorteile*: Einfach, alternative Möglichkeit zu anderen Scans
  - *Nachteile*: Echo-Service auf Geräten meist nicht aktiviert bzw. von Firewalls blockiert

DNS-basierte Methoden:

- **Lookup/Reverse Lookup**
  - *Eingabe*: IP-Adresse bzw. Netzwerkname, (DNS-Server)
  - *Ausgabe*: Netzwerkname bzw. IP-Adresse
  - *Vorteile*: Standardfunktionalität, liefert Hinweise auf möglicherweise aktive Geräte
  - *Nachteile*: Alias-Namen über Reverse Lookup nicht bestimmbar
- **Zone Transfer**
  - *Eingabe*: DNS-Server
  - *Ausgabe*: Alle Daten des Servers (inkl. IP-Adressen, Netzwerknamen und Alias-Namen)

  - *Vorteile*: Keine zu untersuchenden IP-Adressen als Eingabedaten notwendig, liefert alle Daten auf einmal, alle Alias-Namen zu einer IP-Adresse können extrahiert werden
  - *Nachteile*: Funktioniert meist nur zwischen DNS-Servern, möglicherweise keine Erlaubnis für Zone Transfer zur Discovery-Maschine

### 3.12.2 Übersicht nach Art der Information

Folgende Methoden können zum Finden von Geräten (*Host Discovery*) eingesetzt werden:

- *SNMP Scan*
- *AFTs* auslesen
- *ARP Tables* auslesen
- *ARP Ping* (mit/ohne VLAN getaggten Frames)
- *ICMP Ping* (Unicast/Broadcast)
- *TCP ACK Scan*
- *TCP SYN Scan*
- *TCP Echo*
- *UDP Scan*
- *UDP Echo*
- *DNS Reverse Lookup*
- *DNS Zone Transfer*

Folgende Methoden können zum Bestimmen der Layer 3 Topologie (*Layer 3 Discovery*) eingesetzt werden bzw. dafür nötige Teilinformationen liefern:

- *Interface Tables* auslesen
- *Routing Tables* auslesen
- *Traceroute*

Folgende Methoden können zum Bestimmen der Layer 2 Topologie (*Layer 2 Discovery*) eingesetzt werden bzw. dafür nötige Teilinformationen liefern:

- *Interface Tables* auslesen

- *AFTs* auslesen
- *ARP Tables* auslesen
- *VLAN Tables* auslesen
- *STP Tables* auslesen

Die in diesem Kapitel vorgestellten Auflistungen sollen als Leitfaden bei der den individuellen Bedürfnissen entsprechenden Auswahl der passenden Methoden für die gewünschten Aufgaben dienen.

# Kapitel 4

# Der Discovery-Algorithmus

In diesem Kapitel werden einige der zuvor präsentierten Techniken ausgewählt und in einem konkreten Algorithmus miteinander kombiniert. Dieser soll zeigen, wie ein Discovery-System aufgebaut werden kann. Es ist nicht das Ziel dieses Algorithmus', alle möglichen Methoden zu verwenden.

## 4.1 Grundlagen

Der Algorithmus verfolgt folgende Ansätze (vgl. Abschnitt *3.2 Ansätze*): Er ist *aktiv*, das heißt es werden Daten für das Discovery über das Netzwerk gesendet und zwar einerseits um nach Geräten zu suchen (*Probing*) sowie andererseits um Informationen von Infrastrukturkomponenten abzufragen (*Polling*). Weiters ist er für den *zentralen* Einsatz von einer Discovery-Maschine aus konzipiert. Er könnte jedoch prinzipiell auch verteilt verwendet werden, wobei aber dann die an verschiedenen Stellen gesammelten Daten konsolidiert und Strategien zum Umgang mit Diskrepanzen überlegt werden müssten. Außerdem ist der Algorithmus für Ad-hoc-Scans und nicht für eine kontinuierliche Überwachung ausgelegt. Weiters liegt der Fokus des Algorithmus' auf dem Durchsuchen eines zusammenhängenden LANs, welches unter einer einheitlichen Verwaltung steht.

Eine zentrale Stellung im Algorithmus nimmt das Auslesen von Daten über SNMP ein, weil dies einerseits im Network Management weit verbreitet eingesetzt wird und andererseits nur so viele Informationen (ohne übermäßigen Aufwand) erfasst werden können. Dies hat zur Folge, dass die im Netz verwendeten SNMP Communities beim Discovery bekannt sein müssen. Der vorgestellte Algorithmus soll bei der Administration von Netzwerken behilflich sein und weder zum Cracken, noch zum Ausspionieren eines Netzes dienen. Daher wird davon ausgegangen, dass dem Anwender (einem Netzwerk-Administrator) die entsprechenden Informationen (wie etwa SNMP Communities) auch bekannt sind.

Bei der Entwicklung des Algorithmus' stand der praktische Nutzen besonders im Vordergrund und weniger ein Beweis der theoretischen Korrektheit und Vollständigkeit unter bestimmten Umständen – für welche unrealistische Annahmen (z.B. korrekte und vollständige AFT-Informationen, wie in [45]) hätten gemacht werden müssen bzw. die wichtige Spezialfälle (z.B. virtuelle Router) unberücksichtigt gelassen hätten. Weiters hat die Erfahrung bei der Entwicklung dieses Algorithmus' und des ihn umsetzenden Programms gezeigt, dass Robustheit ein entscheidender Punkt ist, da sich beispielsweise einzelne Geräte sehr unterschiedlich verhalten sowie fehlende oder falsche Informationen vorliegen können.

Für den Algorithmus wurden eigene Ideen und Erfahrungen (vor allem was gewisse Feinheiten und Sonderfälle betrifft) mit Vorgehensweisen aus bestehender Literatur (wie im vorigen Kapitel dargestellt) kombiniert.

## 4.2 Voraussetzungen

Um den Algorithmus erfolgreich einsetzen zu können, müssen zuvor einige Voraussetzungen erfüllt sein bzw. Einschränkungen beachtet werden. Das war notwendig, um den Umfang und die Komplexität in Grenzen zu halten, da es sich dabei nur um ein Beispiel handeln soll (allerdings voll funktionsfähig und praktisch einsetzbar). Folgende Rahmenbedingungen gilt es zu berücksichtigen:

- Das Discovery erzeugt einen „Schnappschuss" des Netzwerkes zum „Zeitpunkt" (obwohl es sich eigentlich um eine Zeitspanne handelt) von dessen Ausführung. Währenddessen nimmt der Algorithmus den Zustand des Netzes als statisch an, d.h. es wird keine Rücksicht auf mögliche, aber unwahrscheinliche Änderungen (z.B. der Topologie) während des Discovery-Prozesses genommen. Sollten diese auftreten, könnten die Ergebnisse verfälscht werden. Diese Einschränkung liegt auch bei praktisch allen anderen in der Literatur vorgestellten Algorithmen vor. Weiters wird davon ausgegangen, dass die von Geräten (z.B. über SNMP) angebotenen Daten korrekt in dem Sinne sind, als dass sie den aktuellen Status des Netzwerkes widerspiegeln.

- Der Fokus des Algorithmus' liegt, wie erwähnt, auf einem abgegrenzten, unter einheitlicher Verwaltung stehenden LAN. Es wird daher von konsistent definierten VLANs und dergleichen ausgegangen.

- Die vom Algorithmus eingesetzten Methoden sind auf Netzwerke ausgerichtet, die *Ethernet* sowie *TCP/IP* (und dabei *IPv4*) verwenden, da dies die dominanten Technologien in ihrem Bereich sind.

- Da das Auslesen von Informationen über SNMP eine zentrale Rolle im Algorithmus spielt, muss dieses (vor allem von Routern und Switches) unterstützt werden (einerseits SNMP an sich sowie andererseits die konkret benötigten MIBs).
- Die im Netzwerk verwendeten SNMP Communities müssen bekannt sein (vgl. *4.3 Ein- und Ausgabedaten*), damit der SNMP-Zugriff auch erfolgen kann.
- Weiters müssen von der Discovery-Maschine stammende SNMP-Anfragen auch beantwortet werden (manchmal antworten Geräte aus Sicherheitsgründen nur auf Pakete, die von bestimmten IP-Adressen stammen).
- Zur Bestimmung der Layer 2 Topologie wird in diesem Algorithmus auf STP-Daten zurückgegriffen. Daher muss das Spanning Tree Protocol nicht nur eingesetzt werden, sondern es müssen auch dessen Daten erfassbar und vollständig sein. Der Algorithmus kann zwar mit Lücken im Spanning Tree umgehen, dann aber korrekte Ergebnisse nicht mehr garantieren.
- Die Discovery-Maschine sollte idealerweise an einem Switch Port angeschlossen sein, der so konfiguriert ist, sodass über ihn Frames in alle VLANs gesendet und aus allen VLANs empfangen werden können. Dies ist zwar keine zwingende Voraussetzung, erlaubt es aber *ARP Ping* (siehe Abschnitt *3.4.1 ARP Ping*) einzusetzen, was die Effektivität des Discoverys erhöht.
- Weiters ist es besser, wenn der im Netzwerk verwendete IP-Adressbereich in kleinere Subnetze partitioniert ist, als wenn ein großes Subnetz (ev. mit vielen Lücken bzw. unbenutzten IP-Adressen) verwendet wird. Dies hat zwar keinerlei Auswirkungen auf die Qualität des Ergebnisses, jedoch auf die Geschwindigkeit der Ausführung.

## 4.3 Ein- und Ausgabedaten

Da der Algorithmus ein Discovery bei minimalen Vorkenntnissen ermöglichen soll, sind als einziges Eingabedatum die im Netzwerk verwendeten SNMP Communities zwingend erforderlich.

Zum Erkennen der Infrastruktur wird jedoch auch ein initiales Subnetz benötigt, in welchen die Suche begonnen wird (siehe Abschnitt *4.5 Infrastructure Discovery*). Dieses kann jedoch automatisch aus der IP-Adresse und Subnetzmaske des Interfaces der Discovery-Maschine bestimmt werden. Einzig für den Fall, dass Router nicht über alle ihre Interfaces auf

SNMP-Anfragen antworten, sondern beispielsweise nur über je ein Interface in einem separaten Management-Subnetz, muss dieses initiale Subnetz explizit angegeben werden (so die Discovery-Maschine nicht bereits in diesem liegt). Ansonsten würde die vom Algorithmus verwendete Methode, sich von einem Subnetz zu nächsten vorzuarbeiten, nicht funktionieren, weil man dann nicht über das eigene Subnetz hinauskommen würde.

Alle weiteren Eingabedaten für den Algorithmus sind optional, wie etwa zusätzliche, explizit definierte Subnetze und VLANs zum Durchsuchen (oder umgekehrt zum Ignorieren). Weiters können für einzelne Module bzw. Schritte des Algorithmus' Konfigurationsparameter festgelegt werden (z.B. Timeouts und Retry Counts für SNMP-Operationen, Anzahl an zu benutzenden Threads, das zu verwendende Interface der Discovery-Maschine etc.). Derartige Optionen sind jedoch mehr ein Teil einer konkreten Implementierung (vgl. Abschnitt *B.4.1 Explorer*) und weniger einer abstrakten Anleitung für das generelle Vorgehen.

Als Ausgabe (vgl. auch Abschnitt *5.4 Datenmodell*) liefert der Algorithmus im Netzwerk gefundene, aktive Geräte mit Informationen zu diesen (Interfaces, MAC-Adressen, IP-Adressen, Netzwerknamen, VLANs). Weiters werden die Layer 3 (Subnetze, Router und Routen) sowie die Layer 2 Topologie (VLANs, Switches, Hubs und Verbindungen) bestimmt.

## 4.4 Übersicht

Der Discovery-Algorithmus gliedert sich in fünf Phasen, wie in *Abbildung 4.1* dargestellt, auf welche in den folgenden Abschnitten näher eingegangen wird. Dabei wird jedoch nicht jeder Detailschritt (etwa in Form von Pseudocode) angeführt, sondern es werden die wesentlichen auszuführenden Tätigkeiten (der leichteren Verständlichkeit wegen in Prosa) derart vorgestellt, sodass sich diese in einem Programm konkret umsetzen lassen. Wo es nötig ist, werden jedoch auch Feinheiten betrachtet, weil diese in manchen Fällen ausschlaggebend sind.

Die Phasen des Algorithmus' sind:

1. *Initialization*: Vor dem eigentlichen Discovery gilt es, diverse Vorbereitungen zu treffen (z.B. Variablen zu initialisieren, Datenbank zurückzusetzen etc.). Dies ist zwar konzeptuell ein eigener Schritt, jedoch stark von der konkreten Implementierung abhängig und wird daher hier nicht näher betrachtet.

2. *Infrastructure Discovery*: Das Discovery beginnt, indem die Netzwerk-Infrastruktur (Subnetze, VLANs, Router und Switches) bestimmt

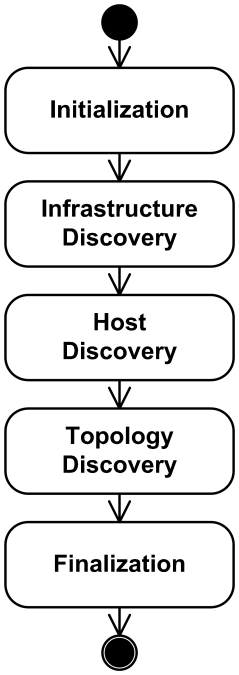


**Abbildung 4.1:** Phasen des Discovery-Algorithmus'

wird. Da nach diesem Schritt Subnetze, Router und Routen bekannt sind, ist die Layer 3 Topologie bereits ausreichend definiert. Weiters werden – sozusagen als Nebeneffekt – auch alle anderen SNMP-Geräte erkannt.

3. *Host Discovery*: In dieser Phase werden die gefundenen Subnetze und VLANs nach aktiven (End-)Geräten durchsucht, wobei verschiedene Techniken eingesetzt werden, um möglichst viele Maschinen zu finden.

4. *Topology Discovery*: Anschließend werden sowohl zusätzliche Informationen von den Switches ausgelesen, als auch alle bisher bestimmten Daten zusammengefügt, um daraus die Layer 2 Topologie zu errechnen.

5. *Finalization*: Am Ende müssen die gesammelten Informationen persistiert (z.B. in einer Datenbank gespeichert) werden. Auch dies hängt von der konkreten Implementierung ab, daher wird hier nicht näher darauf eingegangen.

Einige Aspekte gelten gleichermaßen für alle bzw. mehrere Schritte des Algorithmus' und werden hier betrachtet. Sollen bestimmte Subnetze bzw. IP-Adressbereiche oder VLANs ignoriert (d.h. nicht durchsucht) werden, so ist dies bei den einzelnen Schritten zu berücksichtigen, und die entsprechenden Adressen bzw. VLANs sind zu überspringen. Dies gilt jedoch allgemein und

überall gleichermaßen und wird daher nicht jedes Mal aufs Neue erwähnt. Wenn der im Netzwerk verwendete IP-Adressbereich auf mehrere kleinere Subnetze aufgeteilt ist (anstatt in einem großen Subnetz zu liegen), bietet es sich an, Subnetze auszulassen, die eine bestimmte (definierbare) Größe überschreiten, um so sicherzustellen, dass der Algorithmus beim Durchsuchen den lokalen Adressbereich nicht verlässt. Dies ist entsprechend auf alle Schritte anzuwenden, bei denen Subnetze abgearbeitet werden.

Im Folgenden werden die drei eigentlichen Phasen des Discoverys näher beleuchtet. Bei der grafischen Darstellung der jeweilige Abläufe werden sowohl Aktivitäten (Rechtecke mit runden Ecken), Daten (Rechtecke), der Kontrollfluss (durchgezogene Linien) und der Datenfluss (strichlierte Linien) gezeigt. Bei der Beschreibung liegt der Fokus auf den einzelnen logischen Schritten, die auszuführen sind. In einer konkreten Implementierung könnten Schritte unter Umständen zusammengefasst oder umgeordnet, sowie andere Techniken zur Steigerung der Effizienz und Effektivität eingesetzt werden (z.B. eine Parallelisierung oder mehrere Scan-Durchgänge). Der Übersicht halber werden diese hier jedoch ausgespart.

## 4.5 Infrastructure Discovery

Das Discovery beginnt mit dem *Infrastructure Discovery*. Dessen Ablauf ist in *Abbildung 4.2* dargestellt. Ziel dieser Phase ist es, Subnetze und VLANs sowie Router und Switches zu bestimmen, damit mit Hilfe dieser Informationen in weiterer Folge nach Geräten gesucht und die Topologie bestimmt werden kann.

Das Infrastructure Discovery benötigt als Eingabe zumindest ein initiales Subnetz. Dieses lässt sich, wie erwähnt, automatisch aus der IP-Adresse und Subnetzmaske des Interfaces der Discovery-Maschine bestimmen. Zusätzlich können aber auch explizit weitere initiale Subnetze angegeben werden.

Als Nächstes wird ein Subnetz ausgewählt und nach SNMP-Geräten durchsucht (*SNMP Discovery*). Danach werden von jedem gefundenen Gerät Informationen ausgelesen (*SNMP Information Retrieval*). Daraus ergibt sich einerseits, welche Router und Switches existieren, andererseits werden weitere vorhandene Subnetze (über die Interfaces der SNMP-Geräte) gefunden. Diese Subnetze werden der Liste der bekannten Subnetze hinzugefügt.

Der zuvor genannte Vorgang wird nun so lange für jedes Subnetz wiederholt, bis keine neuen Subnetze mehr gefunden werden und alle bekannten Subnetze durchsucht wurden. Danach werden die Daten der SNMP-Geräte miteinander verglichen, um Maschinen zusammenzuführen, die unter mehreren IP-Adressen und/oder SNMP Communities mittels SNMP erreichbar

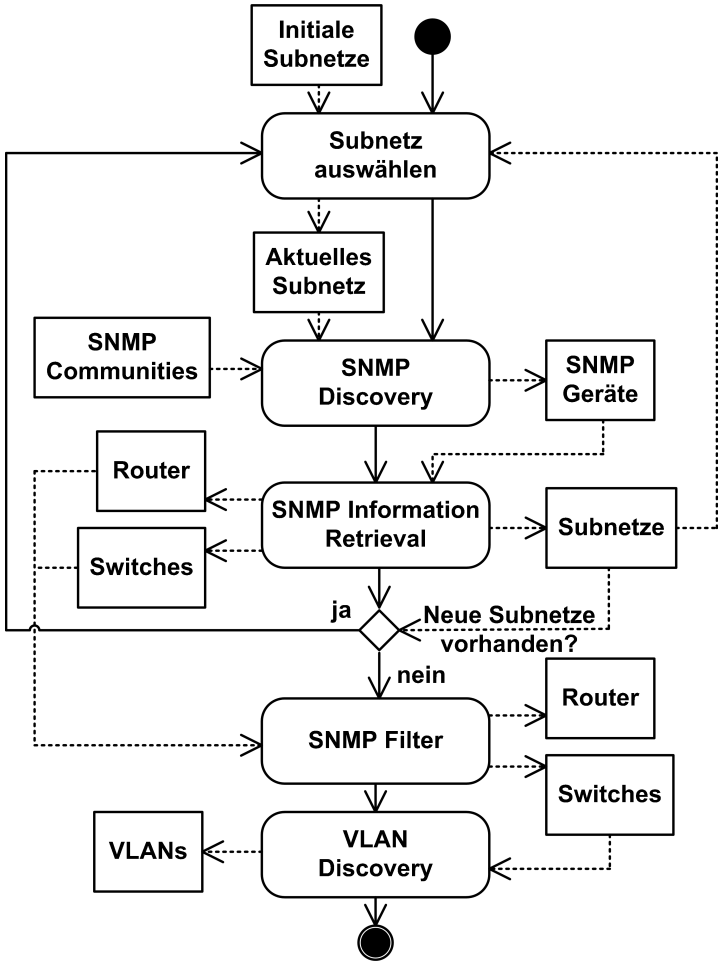


**Abbildung 4.2:** Infrastructure Discovery

sind und somit vorerst als separate Geräte erkannt wurden. Nun gilt es, diese zusammenzufassen (*SNMP Filter*).

Abschließend werden von allen Switches die auf ihnen definierten VLANs ausgelesen (*VLAN Discovery*). Diesen gefundenen VLANs können explizit als Eingabedatum angegebene VLANs hinzugefügt werden.

### 4.5.1 SNMP Discovery

Beim *SNMP Discovery* wird ein Subnetz nach SNMP-Geräten durchsucht (vgl. Abschnitt *3.3.1 SNMP Scan*). Dieses Subnetz sowie alle durchzuprobierenden SNMP Communities werden dazu als Eingabedaten benötigt. Als Ausgabedaten werden Paare von je einer IP-Adresse und einer SNMP Community geliefert, unter denen eine Antwort erhalten wurde.

Konkret geht das SNMP Discovery so vor, dass an jede IP-Adresse aus dem gegebenen Subnetz für jede zu versuchende SNMP Community (d.h. alle Adressen des Subnetzes werden ein Mal pro Community durchlaufen) eine `Get Request PDU` geschickt wird, die nach dem `sysName` fragt (diese Variable sollte von allen SNMP-Geräten unterstützt werden). Netzwerk- und Broadcast-Adressen werden übersprungen, damit nicht mehrere Geräte auf eine Anfrage antworten und so der Umgang mit ihnen verkompliziert wird.

In jedem SNMP-Paket wird die aktuelle SNMP Community (des dazugehörigen Durchlaufs) gesetzt. Zusätzlich wird in dem `Request ID` Feld die jeweilige Ziel-IP-Adresse eingetragen. Dieses Feld kann gemäß dem Standard eine beliebige Nummer enthalten, die es dem Sender erlaubt, Antworten der entsprechenden Anfrage zuzuordnen. Da es keine Regeln bezüglich der Vergabe dieser Nummer gibt, kann dem Feld auch eine IP-Adresse (interpretiert als 4 Byte lange Zahl) zugewiesen werden.

Die Pakete an alle Geräte können in kurzer Abfolge verschickt werden. Erst nach dem letzten Paket muss gewartet werden, um den Geräten auch genug Zeit zum Antworten zu geben. Gleichzeitig wird während des gesamten Vorgangs nach Rückmeldungen gelauscht. Diese Antworten müssen analysiert werden. Einerseits gilt es, nur `Get Response PDUs` weiter zu behandeln (manche Geräte senden etwa von sich aus Nachrichten, und sollten diese empfangen werden, müssen sie verworfen werden). Andererseits muss sichergestellt werden, dass es sich auch um die Antwort auf die richtige Frage handelt, das Paket also nur genau den `sysName` enthält. Auch die SNMP Community muss aus dem Paket extrahiert werden, da nicht alle Geräte mit der gleichen Community antworten, wie mit der sie gefragt wurden, oder es kann eine Antwort auch so verspätet eintreffen, dass bereits ein Scan mit einer anderen Community läuft. Durch den Vergleich der Absenderadresse und der im `Request ID` Feld kodierten IP-Adresse lässt sich bestimmen, ob ein Gerät mit einer anderen IP-Adresse antwortet, als mit der es gefragt wurde.

Am Ende dieses Vorganges sollte von allen im gegebenen Subnetz vorhandenen SNMP-Geräten bestimmt worden sein, unter welchen IP-Adressen und dazu passenden SNMP Communities sie erreichbar sind. Falls ein Gerät unter mehreren IP-Adressen und/oder Communities antwortet (und dabei gleiche oder andere Daten liefert), so werden alle Kombinationen aus IP-Adresse und Community vorerst als jeweils ein eigenständiges Gerät erkannt. Diese werden später (siehe Abschnitt *4.5.3 SNMP Filter*) zusammengefasst.

### 4.5.2 SNMP Information Retrieval

Nachdem bestimmt wurde, welche Geräte im aktuellen Subnetz prinzipiell SNMP-fähig sind und unter welcher IP-Adresse und SNMP Community diese antworten, können von diesen in weiterer Folge spezifisch Daten abgefragt werden. Von allen im SNMP Discovery gefundenen SNMP-Geräten werden folgende Informationen eingeholt:

- Die Basiseigenschaften des Geräts (`sysName`, `sysDescr`, `sysLocation`, `sysContact` und `sysObjectID`) werden abgerufen (vgl. Abschnitt *3.3.2 Geräteinformationen*).
- Es wird festgestellt, ob das Gerät ein Router ist.
- Es wird bestimmt, ob es sich bei dem Gerät um einen Switch handelt.
- Falls das Gerät ein Switch ist, wird festgestellt, ob dieser die `Q-BRIDGE-MIB` unterstützt oder *Cisco VTP* verwendet.
- Wird vom Gerät die `PRINTER-MIB` unterstützt, so wird es als Drucker gekennzeichnet.
- Die Interfaces des Geräts werden bestimmt, indem das `ifTable` ausgelesen wird (vgl. Abschnitt *3.3.3 Interface Tables*). Interfaces mit leerer oder ungültiger (`00:00:00:00:00:00`) MAC-Adresse werden ignoriert, es sei denn mit ihnen ist eine IP-Adresse assoziiert (siehe Punkt über das Auslesen der IP-Adressen). Zwischen physischen und virtuellen Interfaces wird nicht unterschieden.
- Falls ein Gerät ein Funk-Interface besitzt, wird es als WLAN Access Point klassifiziert. Geräte mit Point-to-Point Interfaces werden analog dazu als WAN Gateways markiert. Dies ist jedoch nur eine heuristische Einteilung von Geräten.
- Die Daten über die Interfaces werden mit den Informationen aus dem `ifXTable` ergänzt, falls dieses unterstützt wird.
- Die IP-Adressen des Geräts werden inklusive der dazugehörigen Subnetzmasken aus dem `ipAddrTable` ausgelesen und den Interfaces zugeordnet. Lokale und ungültige IP-Adressen (`127.0.0.1` und `0.0.0.0`) werden übersprungen.
- Von Routern werden zusätzlich die auf ihnen definierten Routen aus dem `ipRouteTable` abgefragt (vgl. Abschnitt *3.3.5 Routing Tables*). Die Default Route (ins „Subnetz“ `0.0.0.0`) wird ignoriert. Die IP-Adresse des Next Hops in indirekten Routen gibt zusätzlich den Hinweis, dass es unter dieser Adresse höchstwahrscheinlich ein aktives Gerät gibt (d.h. auch über die Routing Tables können Geräte gefunden werden).

Nachdem die zuvor beschriebenen Informationen von allen SNMP-Geräten im aktuellen Subnetz abgerufen wurden, wurden damit alle darin vorhandenen Switches und Router erkannt sowie – sozusagen als Nebeneffekt – auch Daten über alle anderen SNMP-Geräte bestimmt. Von den gefundenen Switches und Routern werden in späteren Schritten weitere Informationen ausgelesen.

Außerdem wurden in diesem Schritt weitere vorhandene und möglicherweise noch zu durchsuchende Subnetze ermittelt. Diese werden einerseits von den IP-Adressen und dazugehörigen Subnetzmasken der Geräte sowie andererseits direkt von den Zielsubnetzen in den Routing Tables geliefert.

Da mit diesem Vorgang Router, Routen und Subnetze bestimmt wurden, ergibt sich daraus bereits implizit die Layer 3 Topologie. Diese ist vollständig, nachdem der Schritt für die SNMP-Geräte in allen Subnetzen ausgeführt wurde.

Manchmal ist ein Gerät über SNMP unter einer gewissen IP-Adresse erreichbar, diese scheint dann jedoch nicht in dessen `ipAddrTable` auf. In diesem Fall erzeugt der Algorithmus ein leeres Interface (d.h. ohne MAC-Adresse) und weist diesem die betreffende (Management-)IP-Adresse zu. Dadurch ist sichergestellt, dass alle Informationen über die IP-Adressen von Geräten gespeichert werden – auch in Fällen wie diesem, wo die Interface Tables offensichtlich unvollständige Daten enthalten.

### 4.5.3 SNMP Filter

Da SNMP-Geräte unter mehreren IP-Adressen und/oder SNMP Communities erreichbar sein können, gilt es, *Aliasse* von jeweils ein und demselben Gerät zu bestimmen. Eingabedaten dieses Schritts sind alle gefundenen SNMP-Geräte, Ausgabedaten die gefilterten Geräte (d.h. Aliasse wurden entfernt).

Hier wird ein Gerät als ein Alias von einem anderen Gerät definiert (beide wurden im SNMP Discovery gefunden und haben verschiedene IP-Adressen und/oder Communities), wenn beide (im SNMP Information Retrieval) exakt die *gleichen* Daten geliefert haben. Wenn ein Gerät beispielsweise unter zwei verschiedenen Communities *unterschiedliche* Daten liefert (wie es etwa bei virtuellen Routing Engines auf Switches der Fall ist), so wird dies weiterhin als zwei separate Geräte und *nicht* als Aliasse angesehen.

Ob ein Gerät ein Alias eines anderen ist, kann beispielsweise so festgestellt werden, indem die Daten aller Geräte (exklusive IP-Adresse und SNMP Community) in jeweils einen einzelnen String geschrieben und danach alle

Strings miteinander verglichen werden. Haben zwei Geräte den gleichen Daten-String, so sind sie Aliasse.

Gibt es mehrere Aliasse von einem Gerät, so wird einer davon als „Hauptgerät“ bestimmt und die anderen Aliasse diesem zugeordnet, wobei von diesen nur die entsprechende IP-Adresse und SNMP Community (aber nicht alle anderen Daten) gespeichert werden müssen. Die zuvor gefundenen Geräte, welche sich nun als Aliasse herausgestellt haben, werden gelöscht.

### 4.5.4 VLAN Discovery

In diesem Schritt werden alle definierten VLANs bestimmt. Dabei werden von allen zuvor gefundenen Switches (welche die Eingabedaten darstellen) die nötigen Informationen, wie in Abschnitt *3.3.8 VLAN Tables* beschrieben, abgefragt. Zusätzlich zur Bestimmung der vorhandenen VLANs wird auch eruiert, welcher Switch Port in welchem VLAN liegt (in dem Sinn, welches Default VLAN diesem zugeordnet ist).

## 4.6 Host Discovery

Aufgabe der nächsten Phase – dem *Host Discovery* – ist es, die im Infrastructure Discovery bestimmten Subnetze und VLANs nach Geräten zu durchsuchen. Dazu werden verschiedene Techniken verwendet, um möglichst viele Informationen zu gewinnen (wobei die Reihenfolge der Teilschritte nicht relevant ist). Dies ist in *Abbildung 4.3* dargestellt.

Das *ARP Table Discovery* liest die ARP Tables der Router aus und findet so IP- und MAC-Adressen von Geräten. Beim *ARP Discovery* werden ARP Requests für alle IP-Adressen aus allen Subnetzen in alle VLANs gesendet und so auch IP- und MAC-Adressen von aktiven Geräten bestimmt. Das *ICMP Discovery* pingt alle Geräte im vorgegebenen Adressbereich und findet damit ebenfalls aktive Geräte (dabei aber nur deren IP-Adresse). Schließlich führt das *DNS Discovery* einen Reverse Name Lookup für alle IP-Adressen durch und bestimmt so die dazugehörigen DNS-Namen.

Bei der Umsetzung der hier beschriebenen Teilschritte muss darauf geachtet werden, Daten einander richtig zuzuordnen. Wurde beispielsweise in einem Schritt nur die IP-Adresse eines aktiven Geräts gefunden (z.B. im ICMP Discovery), in einem zweiten nur eine MAC-Adresse (z.B. beim Auslesen einer Interface-Tabelle im SNMP Information Retrieval) und in einem dritten die Zuordnung von der IP- zur MAC-Adresse (z.B. im ARP Discovery), so muss sichergestellt werden, dass diese Informationen zu einem Gerät konsolidiert werden und nicht etwa drei separate generiert werden.

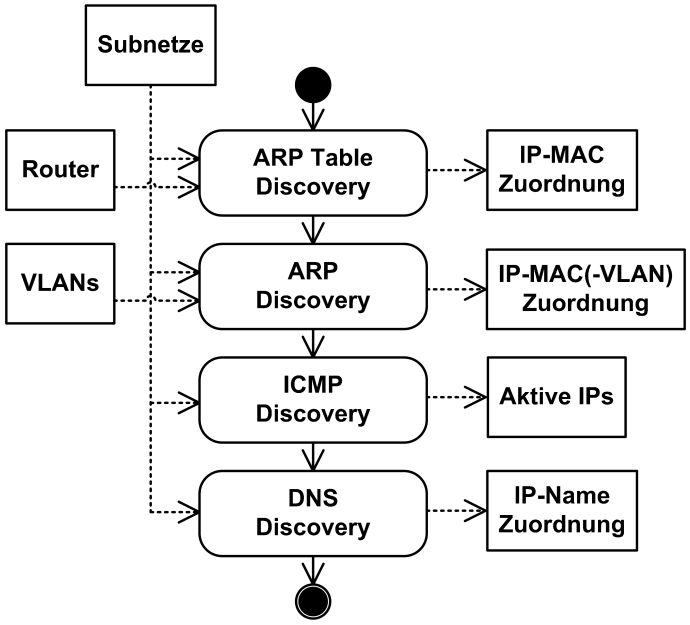


**Abbildung 4.3:** Host Discovery

Weiters muss bei der Zuordnung von IP-Adressen zu Subnetzen vorsichtig vorgegangen werden, falls fehlerhaft überlappend definierte Subnetze vorhanden sind (z.B. `192.168.0.0/24` und `192.168.0.0/16`), denn in diesem Fall könnte eine IP-Adresse (z.B. `192.168.0.1`) zu mehreren Subnetzen gehören. Eine Möglichkeit damit umzugehen ist (neben dem Ausgeben einer Fehlermeldung), die Adressen jeweils mit dem kleinsten Subnetz (in diesem Beispiel `192.168.0.0/24`) zu assoziieren.

### 4.6.1 ARP Table Discovery

Die Aufgabe des *ARP Table Discoverys* ist, wie schon der Name sagt, die ARP Tables von Routern auszulesen und so aktive Geräte *indirekt* zu erkennen. Eingabedaten sind die vorhandenen Subnetze sowie die Router in jedem Subnetz. Ausgabedaten sind die IP- und MAC-Adressen der Geräte.

Alle vorhandenen Subnetze werden in mehreren Durchgängen – einer pro Subnetz – abgearbeitet. Da Einträge in ARP Tables verhältnismäßig kurzlebig sind, werden die Tabellen durch das Senden von Pings (vgl. Abschnitt *3.6.1 Ping*) zuerst befüllt, bevor sie ausgelesen werden. Danach werden die Tabellen von jedem Router im aktuellen Subnetz ausgelesen, genau wie in Abschnitt *3.3.7 ARP Tables* gezeigt.

### 4.6.2 ARP Discovery

Beim *ARP Discovery* werden ARP Requests (mit und ohne VLAN Tags) verschickt (wie in Abschnitt *3.4.1 ARP Ping* beschrieben), um damit aktive Geräte direkt zu finden. Die zu durchsuchenden Subnetze und VLANs sind die Eingabedaten dieses Schritts, Ausgabedaten sind die IP- und MAC-Adresse jedes gefundenen Geräts sowie eventuell das VLAN in dem es liegt.

Es wird ein ARP Request für jede IP-Adresse aus jedem Subnetz verschickt, wobei dieser Vorgang in mehreren Durchgängen wiederholt wird. In jedem Durchlauf wird ein anderes VLAN Tag an die Frames angehängt. Dadurch werden alle VLANs durchsucht und die Antworten können auch dem entsprechenden VLAN zugeordnet werden. Zusätzlich gibt es einen weiteren Durchgang, bei dem normale Frames ohne VLAN Tags verschickt werden. Damit kann zwar nur das lokale VLAN gescannt werden, dies funktioniert dafür aber auch, wenn das Senden und Empfangen von VLAN getaggten Frames nicht möglich ist (beispielsweise weil der Switch Port, an dem die Discovery-Maschine hängt, nicht entsprechend konfiguriert ist) oder wenn die vorhandenen VLANs nicht bestimmt werden konnten.

Die ARP Requests können schnell hintereinander verschickt werden, erst nach dem letzten Frame muss etwas gewartet werden, um den Geräten Zeit zum Antworten zu geben. Nebenbei wird nach deren ARP Replies gelauscht. Wird ein derartiger empfangen, so wurde ein aktives Gerät gefunden, dessen IP- und MAC-Adresse (und eventuell auch VLAN) nun bekannt sind.

### 4.6.3 ICMP Discovery

Das Ziel des *ICMP Discoverys* ist es, aktive Geräte direkt über das Senden von Pings zu finden, wie in Abschnitt *3.6.1 Ping* erläutert. Eingabedaten sind die zuvor gefundenen Subnetze, Ausgabedaten die darin liegenden IP-Adressen von aktiven Geräten (d.h. von denen eine Antwort empfangen wurde).

An jede IP-Adresse aus jedem Subnetz wird ein ICMP Echo Request gesendet. Während des ganzen Vorganges wird nach Rückmeldungen gelauscht. Wird ein ICMP Echo Reply von einem Gerät empfangen und stimmen die darin enthaltenen Daten mit den hingesendeten überein, so wird die entsprechende IP-Adresse als zu einem aktiven Gerät gehörig markiert.

### 4.6.4 DNS Discovery

Beim *DNS Discovery* wird für jede IP-Adresse aus jedem im Infrastructure Discovery gefundenen Subnetz (diese sind die Eingabedaten) ein Reverse Name Lookup durchgeführt, wie in Abschnitt *3.9.1 Lookup und Reverse*

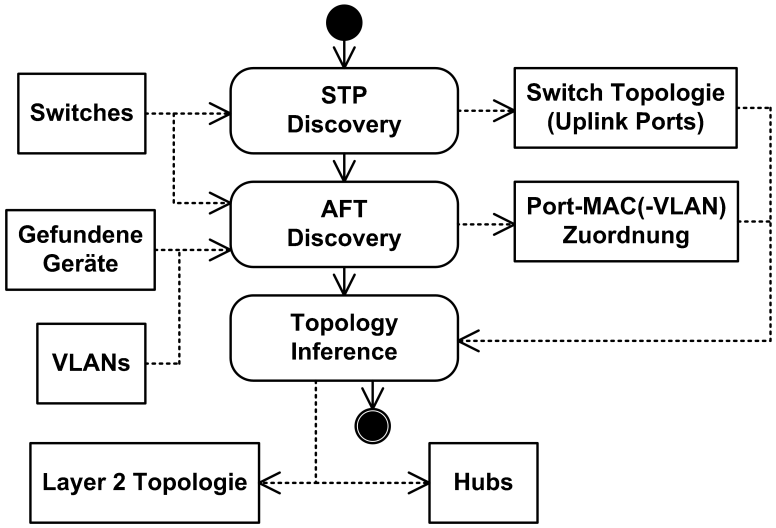


**Abbildung 4.4:** Topology Discovery

*Lookup* beschrieben. Die Ausgabedaten sind daher der zu jeder IP-Adresse gehörige DNS-Name (falls es einen solchen gibt). Alias-Namen können nicht bestimmt werden.

Es wird versucht, ein Lookup für *jede* IP-Adresse (und nicht nur für die Adressen von in vorigen Schritten als aktiv bestimmten Geräten) durchzuführen, da dies zusätzliche Hinweise auf möglicherweise vorhandene Geräte geben kann (auch wenn diese etwa während des Discoverys abgeschaltet sind und daher nicht als aktiv erkannt werden).

## 4.7 Topology Discovery

Die letzte Phase im eigentlichen Discovery ist das *Topology Discovery*, welches in *Abbildung 4.4* dargestellt ist.

Ziel dieser Phase ist die Bestimmung der Layer 2 Topologie (die Layer 3 Topologie ergibt sich bereits implizit aus den zuvor gesammelten Daten). Es wird der Spanning Tree der im Infrastructure Discovery gefundenen Switches bestimmt (*STP Discovery*) sowie deren AFTs ausgelesen (*AFT Discovery*). Mittels dieser Daten kann die Layer 2 Topologie errechnet werden (*Topology Inference*). Zusätzlich werden in diesem letzten Teilschritt Geräte weiter klassifiziert (z.B. virtuelle Router und Maschinen, ARP Proxies, nicht-SNMP Switches etc. erkannt).

### 4.7.1 STP Discovery

Die Aufgabe des *STP Discoverys* ist zu bestimmen, wie die Switches untereinander verbunden sind (d.h. welcher Uplink Port an jeweils welchem anderen angeschlossen ist). Dazu werden die STP-Daten aller Switches, exakt wie in Abschnitt *3.3.9 STP Tables* beschrieben, ausgelesen. Eingabedaten sind die zuvor im Infrastructure Discovery gefundenen Switches.

Zu beachten ist, dass die Kanten im Spanning Tree gerichtet sind, also bei einer Verbindung zweier Switches immer nur ein Uplink Port auf den zweiten verweist, nicht aber umgekehrt. Trotzdem müssen die Uplink Ports beider Switches als solche markiert und deren Zuordnung im Datenmodell beidseitig eingetragen werden.

### 4.7.2 AFT Discovery

Im *AFT Discovery* werden die AFTs aller Switches, ebenfalls genau wie in Abschnitt *3.3.6 Address Forwarding Tables* erläutert, abgefragt, um so zu bestimmen, welche Geräte (identifiziert über die MAC-Adressen ihrer Interfaces) an welchen Switch Ports angeschlossen sind. Eingabedaten sind wiederum die im Infrastructure Discovery gefundenen Switches sowie zusätzlich die definierten VLANs und die im Host Discovery bestimmten Geräte.

Bevor die AFTs ausgelesen werden, gilt es sicherzustellen, dass diese mit ausreichend Daten gefüllt sind. Dazu wird an die jeweilige IP-Adresse aller zuvor im Host Discovery gefundenen Geräte ein Ping (ICMP Echo Request) sowie ein ARP Request (in das entsprechende VLAN, falls bekannt) verschickt. Dabei geht es jedoch nicht darum, dass die Discovery-Maschine die Antworten empfängt, sondern dass diese von den Switches gehört werden.

Danach kann der Inhalt der AFTs aller Switches abgefragt werden. Je nach Switch muss zwischen drei Fällen unterschieden werden:

- Auf dem Switch sind keine VLANs definiert. In diesem Fall wird das `dot1dTpFdbTable` ausgelesen und die Nummern der Ports mittels dem `dot1dBasePortTable` in die entsprechenden Interfaces des Switches übersetzt.
- Auf dem Switch sind VLANs definiert und es wird das *Cisco VTP* verwendet. Auch in diesem Fall werden das `dot1dTpFdbTable` und das `dot1dBasePortTable` ausgelesen, jedoch mehrfach und zwar ein Mal pro definiertem VLAN (aus diesem Grund werden die VLANs als Eingabedatum benötigt), wobei die SNMP Community entsprechend den Regeln des Community Indexings in jedem Durchgang jeweils angepasst wird.

- Auf dem Switch sind VLANs definiert und es wird die `Q-BRIDGE-MIB` unterstützt. In diesem Fall ist das `dot1qTpFdbTable` auszulesen (jedoch nur ein Mal), wobei zusätzlich dessen Einträge mittels des `dot1qVlanCurrentTable` mit dem entsprechenden VLAN korreliert werden müssen. Die Übersetzung der Port-Nummern in Interfaces geschieht weiterhin über das `dot1dBasePortTable`.

In den beiden letzten Fällen kann zu jedem Interface (jeder MAC-Adresse) nicht nur bestimmt werden, mit welchen Port dieses verbunden ist, sondern auch in welchem VLAN es liegt (falls dies nicht schon durch das ARP Discovery bekannt ist). In allen Fällen gilt, dass nur Einträge mit dem `dot1dTpFdbStatus` bzw. `dot1qTpFdbStatus learned` berücksichtigt bzw. Einträge mit ungültigen MAC-Adressen (`00:00:00:00:00:00`) ignoriert werden.

Falls davon ausgegangen wird, dass die AFTs nur eine bestimmte Anzahl von Einträgen enthalten können, es jedoch mehr Geräte als diese im Netzwerk gibt, so kann der Vorgang in einzelne Durchgänge aufgeteilt werden, indem jeweils nur eine gewisse Maximalzahl von Geräte verarbeitet wird. Wird beispielsweise angenommen, dass AFTs maximal $n$ Einträge fassen können, so würden im ersten Durchgang die ersten $n$ Adressen (mit ICMP und ARP) gepingt und dann die AFTs aller Switches ausgelesen. Im zweiten Durchgang würden die nächsten $n$ Adressen gepingt und dann wieder die AFTs von allen Switches abgefragt usw.

### 4.7.3 Topology Inference

Abschließend kann die Layer 2 Topologie aus den zuvor gesammelten Daten errechnet werden (es werden nur bereits vorhandene Informationen verarbeitet und keine neuen mehr eingeholt). Aus den Daten der AFTs (welche MAC-Adresse an welchem Switch Port gehört wurde) sowie der Verbindungen der Switches untereinander (wie durch das STP definiert) werden die Verbindungen aller Interfaces bestimmt sowie, falls nötig, Hubs dazwischen eingefügt – diese können zwar nicht direkt erkannt werden, deren Vorhandensein wird jedoch implizit durch andere Daten angezeigt. Weiters werden in diesen Schritt Geräte weiter klassifiziert und etwa virtuelle Maschinen, virtuelle Router oder ARP Proxies identifiziert.

Die Bestimmung der Layer 2 Topologie läuft in mehreren Schritten ab. Diese werden im Folgenden beschrieben und sollen zusätzlich durch ein Beispiel verdeutlicht werden. Dessen Ausgangslage ist in *Abbildung 4.5* grafisch dargestellt. Es gibt darin jeweils drei Switches und Hosts sowie einen Hub, über den zwei der Hosts mit einem einzigen Switch Port verbunden sind. Die Besonderheit des dritten Hosts ist, dass dieser zwei Interfaces besitzt und

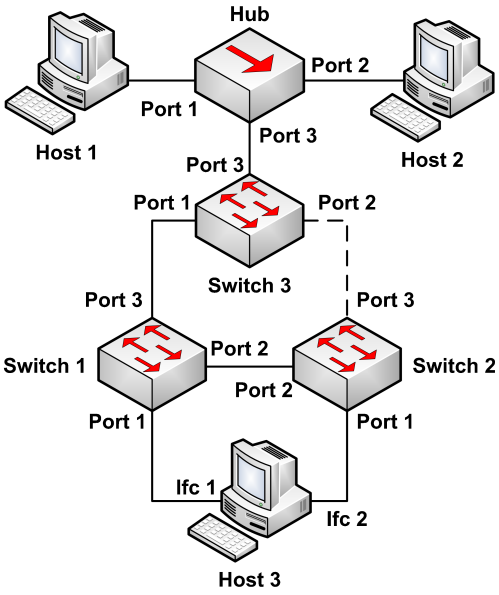


**Abbildung 4.5:** Beispiel-Topologie

über diese mit zwei Switches gleichzeitig verbunden ist, wobei darüber hinaus beide Interfaces die gleiche MAC-Adresse verwenden (dass dies so ist, wurde über das Auslesen dessen Interface Tables bestimmt).

Die Layer 2 Topologie wird nun in den folgenden Schritten errechnet:

1. Basis für die Inferenz der Topologie sind die Daten aus den AFTs. Jeder Switch Port wird mit allen Interface verbunden, deren MAC-Adresse am jeweiligen Port gefunden wurde. *Abbildung 4.6* zeigt diese Verbindungen im Beispiel. Anzumerken ist einerseits, dass Interfaces auch mit den Uplink Ports von Switches assoziiert werden (da auch auf diesen die dazugehörigen MAC-Adressen gehört werden) sowie andererseits beim multi-homed Gerät beide Interfaces mit jeweils beiden Switch Ports verbunden sind, da sich die Interfaces eine MAC-Adresse teilen und nicht unterschieden werden kann, welches tatsächlich wo angeschlossen ist.

2. Als Nächstes werden die STP-Daten miteinbezogen, woraus sich die Verbindungen der Switches untereinander ergeben. Da nun die Uplink Ports der Switches bekannt sind, werden alle Assoziationen dieser mit Interfaces von Endgeräten gelöscht, da angenommen wird, dass Switches immer unmittelbar miteinander verbunden sind und nicht etwa über Hubs. Unter dieser Voraussetzung ist es auch unmöglich, dass Endgeräte an Uplink Ports angeschlossen sind. Deswegen werden die entsprechenden Verbindungen entfernt. Für das Beispiel ist dies in *Abbildung 4.7* dargestellt.

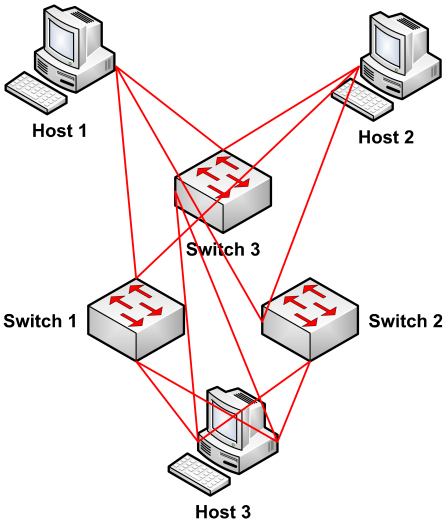


**Abbildung 4.6:** Verbindungen nach den AFT-Daten

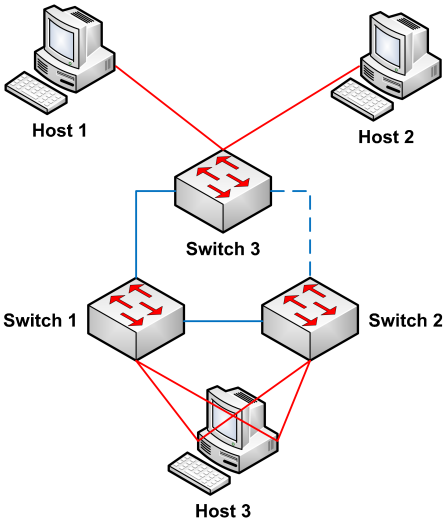


**Abbildung 4.7:** Verbindungen nach Einbeziehung der STP-Daten

3. Zusätzlich werden die Anschlussdaten von Backplane Ports gelöscht, auch wenn sie nicht im Spanning Tree vorkommen und daher nicht explizit als Uplink Ports erkannt wurden. Da Backplane Ports aber generell immer modulare Switches (Slots eines Verteilers) miteinander verbinden (und somit Uplink Ports sind), können Endgeräte nicht direkt daran angeschlossen sein, weshalb die entsprechenden Assoziationen entfernt werden. Backplane Ports werden über eine Heuristik erkannt, nämlich wenn in deren Name oder Beschreibung das Wort `Backplane` vorkommt.

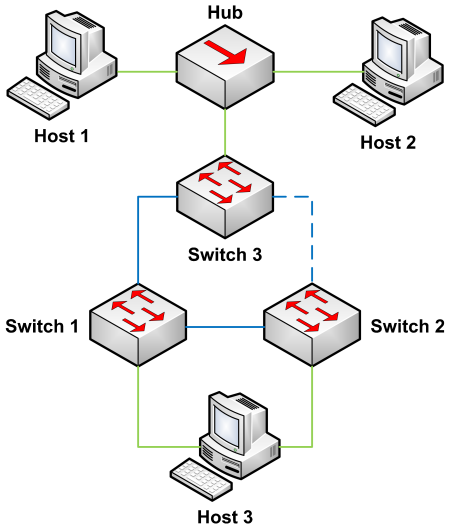


**Abbildung 4.8:** Verbindungen nach dem Einfügen von Hubs

4. Da aber auch nach Einbeziehung der STP-Daten Switch Ports mit mehreren Interfaces oder umgekehrt Interfaces mit mehreren Switch Ports verbunden sein können, gilt es jetzt, diese Situation aufzulösen, sodass jeder Switch Port bzw. jedes Interface immer mit nur jeweils einem Gegenstück assoziiert ist. Generell gilt, dass immer dann, wenn ein Port mit mehreren Interfaces verbunden ist, dazwischen ein Hub eingefügt wird. Alle Interfaces, die zuvor mit diesem Port verbunden waren, werden nun an dem neuen Hub angeschlossen (an je einem eigenen Port des Hubs).

5. Bei multi-homed Geräten, deren Interfaces eine MAC-Adresse mehrfach verwenden (d.h. je ein Interface ist mit mehreren Ports assoziiert), wird eine spezielle Heuristik angewandt. Anstatt alle beteiligen Interfaces und Switch Ports über einen Hub zusammenzuschließen, wird das erste Interface, welches die jeweilige MAC-Adresse verwendet, mit dem ersten Port, auf dem diese MAC-Adresse gehört wurde, verbunden. Das zweite Interface, welches die MAC-Adresse benutzt, wird mit dem zweiten Port assoziiert usw. Gibt es mehrere Interfaces mit der gleichen MAC-Adresse, wurde diese jedoch nur an einem Port gehört, so wird nur das erste Interface (und nicht alle) mit ihm verbunden (außer die anderen Interfaces mit der gleichen MAC-Adresse gehören zu anderen Geräten). Wurde umgekehrt eine MAC-Adresse an mehreren Ports gehört, gibt es aber nur ein Interface mit dieser MAC-Adresse, so werden dem Gerät neue Interfaces – alle mit der entsprechenden MAC-Adresse – hinzugefügt und diese analog zu der zuvor beschriebenen Regel mit den Ports verbunden. *Abbildung 4.8* zeigt den Endzustand der Erkennung der Topologie im Beispiel.

6. Die Regel, dass je ein Interface mit nur je einem Port (von einem Switch oder Hub) verbunden sein darf, gilt nicht für Funk-Interfaces. An diesen können auch mehrere Interfaces „angeschlossen“ sein.

Die Behandlung von mehrfach verwendeten MAC-Adressen ist eine Heuristik, die nicht sicherstellen kann, dass jeweils die richtigen Interfaces und Switch Ports miteinander verbunden werden (so wie sie tatsächlich physisch zusammengeschlossen sind), sondern diese können auch vertauscht sein. Da Interfaces über ihre MAC-Adresse identifiziert werden, gibt es aber keine Möglichkeit diese in diesem Fall zu unterscheiden. Dennoch liefert diese Heuristik in der Praxis sehr gute Ergebnisse, da zumindest erkannt wird, dass ein Gerät mit mehreren Interfaces mehrfach an unterschiedlichen Switches angeschlossen ist (auch wenn unter Umständen Interfaces und Switch Ports einander nicht korrekt zugeordnet werden). Dieser Fall ist typisch für Server mit redundantem Anschluss zur Erhöhung der Betriebssicherheit.

Weiters ist auch die Annahme, dass Switches immer direkt miteinander verbunden sind, dazwischen keine Hubs liegen und daher Endgeräte nie mit Uplink Ports verbunden sein können, eine Heuristik (denn dies muss nicht zwingend so sein, auch wenn eine derartige Verkabelung völlig unüblich ist). In [76] wird eine Vorgehensweise beschrieben, mit der auch Geräte lokalisiert werden können, die zwischen mehreren (über einen Hub verbundenen) Switches liegen. In dem hier vorgestellten Algorithmus wird dies jedoch absichtlich nicht berücksichtigt, da Tests gezeigt haben, dass auf Grund von fehlerhaften Daten – z.B. veralteten AFT-Einträgen, die durch ein zu langes Timeout entstanden sind (vgl. Abschnitt *6.4 Besondere Beobachtungen*) – bei Verwendung eines „genaueren“ Algorithmus im Endeffekt dennoch schlechtere Ergebnisse geliefert werden. Ein solcher ist weniger robust und die Einschränkung, dass zwischen Switches keine Hubs liegen dürfen, wird dagegen eingetauscht, dass die Daten vollständig und korrekt sein müssen.

Da bei dem hier vorgestellten Algorithmus die praktische Anwendbarkeit (im Gegensatz zur völligen theoretischen Korrektheit) im Vordergrund steht, wurde ein einfacheres, aber robusteres Vorgehen gewählt – für diverse kompliziertere Algorithmen zur Topologiebestimmung sei beispielsweise auf die in Abschnitt *1.3 Stand der Forschung* angeführten Arbeiten verwiesen. Einerseits hat es sich gezeigt, dass es wahrscheinlicher ist, dass AFT-Daten falsch, als dass Switches wirklich über Hubs miteinander verbunden sind. Sollte dies andererseits aber entgegen der Erfahrung tatsächlich der Fall sein, so wird die Topologie immer noch korrekt erkannt, einzig die zwischen den Switches liegenden Geräte können nicht genau lokalisiert werden (was der Algorithmus aber bemerken und den Anwender darauf hinweisen kann). Umgekehrt könnte ein „genauerer“ Algorithmus diesen Fall zwar kor-

rekt behandeln, bei fehlerhaften Daten würde er aber eine komplett falsche Topologie generieren – der „Schaden“ wäre also unverhältnismäßig größer.

Wie eingangs erwähnt, wird im Topology Inference Schritt nicht nur die Layer 2 Topologie aus den AFT- und STP-Daten errechnet, sondern auch Geräte weiter klassifiziert. Folgende Erkennungen werden vorgenommen:

- *Nicht-SNMP Switches*: Da in der `BRIDGE-MIB` empfohlen wird, dass die Bridge ID eines Switches gleich der kleinsten auf ihm vorhandenen MAC-Adresse sein soll, ist es so auch möglich, Switches als solche zu erkennen, auch wenn auf sie kein SNMP-Zugriff besteht (etwa weil die nötigen SNMP Communities nicht bekannt sind). Aus der ID der Designated Bridge für jeden Port im Spanning Tree (vgl. Abschnitt *3.3.9 STP Tables*) lässt sich diese MAC-Adresse extrahieren. Wird eine MAC-Adresse von einem im Netzwerk gefundenen Gerät (das nicht über SNMP angesprochen werden kann) auch in den STP-Daten eines Switches (auf welchen SNMP-Zugriff besteht) angetroffen, so ergibt sich daraus, dass das entsprechende Gerät auch ein Switch ist. Dies ist deswegen eine Heuristik, da die Methode nur auf einer Empfehlung und keiner zwingenden Anforderung basiert. Weiters ist es auch unwahrscheinlich, dass die MAC-Adresse eines Switch Ports im Netzwerk gehört wird, so es sich nicht um dessen Management Interface handelt.

- *VMware*: Virtuelle Maschinen vom Typ *VMware* werden anhand ihrer MAC-Adresse als solche erkannt und entsprechend gekennzeichnet (siehe Abschnitt *3.4.2 Virtual Device Detection*).

- *Virtuelle Router*: Virtuelle Router (VRRP und HSRP) werden ebenfalls anhand ihrer MAC-Adresse identifiziert (siehe Abschnitt *3.4.2 Virtual Device Detection*). Informationen bezüglich der Zuordnung von virtuellen und physischen Routern werden zwar nicht über SNMP ausgelesen, stimmt der Standort (in der Layer 2 Topologie) eines virtuellen Routers jedoch mit dem eines physischen Routers überein, so werden diese beiden miteinander verschmolzen. Dadurch können virtuelle Router zumindest demjenigen physischen Router zugeordnet werden, der gerade der Master ist.

- *Virtuelle Routing Engines*: Falls es jeweils einen Switch und einen Router gibt, die über SNMP unter genau den gleichen IP-Adressen, jedoch unter verschiedenen SNMP Communities erreichbar sind, so wird der Router als virtuelle Routing Engine erkannt. Der Router wird weiterhin als separates Gerät verwaltet, jedoch in der Layer 2 Topologie am virtuellen Router Port des Switches angeschlossen. Welcher Port das ist, wird über eine Heuristik bestimmt: Es wird dasjenige Interfa-

ces des Switches genommen, welches die gleiche MAC-Adresse wie der virtuelle Router verwendet.

- *ARP Proxies*: ARP Proxies werden mit einer einfachen Heuristik erkannt. Sind mehr als eine definierbare Anzahl von IP-Adressen mit einer einzelnen MAC-Adresse assoziiert, so wird diese MAC-Adresse als zu einem ARP Proxy gehörig markiert und für jede IP-Adresse wird ein eigenständiges Gerät erzeugt. Gibt es ein SNMP-Gerät, welches laut seiner Interface Table ein Interface mit einer ARP Proxy MAC-Adresse besitzt, so wird das ganze Gerät als ARP Proxy gekennzeichnet.

## 4.8 Zusammenfassung

Abschließend sollen hier die Eckpunkte des Algorithmus' zusammenfassend betrachtet werden, d.h. wie er sich in bestimmten Fällen verhält:

- *Layer 3 Topologie*: Die Layer 3 Topologie (Subnetze, Router und Routen) wird erkannt, und zwar unter der Voraussetzung, dass die Informationen der Router über SNMP ausgelesen werden können.

- *Layer 2 Topologie*: Die Layer 2 Topologie (Switches, Hubs, Geräte, Interfaces und Verbindungen) wird ebenfalls erkannt, sofern die Informationen der Switches über SNMP ausgelesen werden können und diese das Spanning Tree Protocol verwenden.

- *VLANs*: Die definierten VLANs werden von den Switches ausgelesen. Dabei werden sowohl Geräte unterstützt, welche die Informationen über die `Q-BRIDGE-MIB` anbieten, als auch jene, die das *Cisco VTP* verwenden. VLANs werden auch Switch Ports zugeordnet, allerdings wird die komplette VLAN-Konfiguration (z.B. tagged/untagged Ingress und Egress Ports etc.) eines jeden Switches nicht erfasst.

- *Nicht-SNMP Switches*: Switches, auf die über SNMP nicht zugegriffen werden kann, können dennoch erkannt werden, falls in den STP-Informationen anderer Switches (die über SNMP zugänglich sind) auf diese verwiesen wird. Jedoch stellt diese Erkennung einerseits nur eine Heuristik dar, die nicht garantiert werden kann. Andererseits können die an nicht-SNMP Switches angeschlossenen Geräte nicht den entsprechenden Ports zugeordnet werden.

- *Nicht-STP Switches*: Falls manche Switches zwar über SNMP erreichbar sind, aber STP nicht verwenden bzw. keine STP-Daten anbieten, so verfolgt der Algorithmus eine „best effort“ Strategie. Das heißt, es wird versucht, immer wo es möglich ist, ein korrektes Ergebnis zu

liefern – die Korrektheit in jedem Detail kann dabei nicht garantiert werden.

- *Aktive Geräte*: Verschiedene Methoden werden verwendet, um aktive Geräte aufzuspüren. Dazu gehören ICMP Echo und ARP Requests.

- *Firewalls*: Es wird versucht, Firewalls zu umgehen, indem unter anderem mittels ARP Requests nach Geräten gesucht wird.

- *Multi-Homed Geräte*: Falls Geräte mehrere Interfaces haben, werden diese der jeweiligen Maschine richtig zugeordnet, falls sie die Daten über ihre Interfaces über SNMP anbietet (und diese Daten korrekt sind). Interfaces von nicht-SNMP-fähigen multi-homed Geräten werden als separate Geräte erkannt.

- *Redundanter Anschluss*: Sind multi-homed Geräte an verschiedenen Stellen angeschlossen, verwenden dabei aber über mehrere oder alle Interfaces die gleiche MAC-Adresse, werden die Anschlussorte trotzdem korrekt erkannt. Dabei wird davon ausgegangen, dass es sich überall um jeweils das gleiche Gerät handelt und nicht um mehrere Geräte, die falsch konfiguriert sind und die gleiche MAC-Adresse verwenden. Es kann jedoch nicht garantiert werden, dass jedem Anschlussort auch das jeweils korrekte Interface des Geräts zugeordnet wird (d.h. diese können auch vertauscht sein), da sie nicht voneinander unterschieden werden können.

- *Link Aggregation*: Der Algorithmus erkennt, welche Interfaces virtuelle Interfaces einer Link Aggregation Group sind, die dazugehörigen physischen Interfaces werden diesen jedoch nicht zugeordnet.

- *Virtuelle Maschinen*: Virtuelle Maschinen vom Typ *VMware* werden vom Algorithmus mittels der im vorigen Kapitel beschriebenen Heuristik erkannt. Jedoch wird nicht zwischen Hosts und Guests unterschieden und diese auch nicht einander zugeordnet (d.h. auch virtuelle Maschinen werden als eigenständige Geräte behandelt). Dabei kann eine vollständige und korrekte Erkennung nicht garantiert werden.

- *Virtuelle Router*: Adressen von virtuellen Routern (VRRP und HSRP) werden über ihre MAC-Adresse erkannt (jedoch weiterhin als separate Geräte behandelt). Es wird versucht, diese auch mit einem physischen Router zu assoziieren, wobei eine Heuristik verwendet wird, welche die Orte analysiert, an denen virtuelle MAC-Adressen gefunden wurden. Es werden aber keine diesbezüglichen Informationen über SNMP ausgelesen und keine vollständige Zuordnung von physischen und virtuellen Routern vorgenommen.

- *Virtuelle Routing Engines*: Wenn sich SNMP-Geräte unter den selben IP-Adressen, aber mit verschiedenen SNMP Communities alternativ als Switch und Router darstellen, so werden diese einander zugeordnet. Dabei werden jeweils zwei Geräte (ein Switch und ein Router) separat erkannt, deren Ports aber miteinander verbunden (wobei die zu verbindenden Ports über eine Heuristik bestimmt werden). In der Layer 2 Topologie sieht es dann so aus, dass der Router am Switch angeschlossen ist (auch wenn es sich in diesem Fall nur um einen virtuellen Router handelt der physisch auf dem selben Gerät existiert und die Komponenten über virtuelle Ports verbunden sind).

- *Drucker*: Drucker werden als solche erkannt, wenn man auf sie über SNMP zugreifen kann und sie die `PRINTER-MIB` anbieten. Dies ist auch eine Heuristik, da damit sehr viele, aber nicht zwingend alle Drucker erkannt werden können.

- *WLAN Access Points*: Falls Geräte mit einem oder mehreren Funk-Interfaces gefunden werden, so werden diese als WLAN Access Points klassifiziert. Dies ist ebenfalls eine Heuristik, sodass kein vollständiges und hundertprozentig korrektes Ergebnis garantiert werden kann.

- *WAN Gateways*: Analog zum vorigen Punkt werden WAN Gateways erkannt, wenn diese über SNMP angeben, PPP-Interfaces zu besitzen. Auch dies ist eine Heuristik mit den zuvor genannten Konsequenzen.

- *ARP Proxies*: Zur Erkennung von ARP Proxies wird eine einfache Heuristik verwendet, bei der Geräte dann als solche markiert werden, wenn sie MAC-Adressen besitzen, die mit besonders vielen IP-Adressen assoziiert sind.

# Kapitel 5

# Umsetzung des Algorithmus'

Das im Rahmen dieser Arbeit entwickelte Programm *Network Explorer* setzt den im vorigen Kapitel beschriebenen Algorithmus um und wird in diesem näher betrachtet. Auf den Algorithmus selbst bzw. andere Aspekte des Discoverys wird hier nicht mehr eingegangen, da die Software diese exakt wie zuvor beschrieben implementiert. Der Fokus dieses Kapitels liegt darauf zu zeigen, wie ein vorgegebener Algorithmus konkret umgesetzt werden kann.

## 5.1 Übersicht

Der *Network Explorer* ist eine *Java 1.6*[1] Applikation und benötigt daher zur Ausführung eine entsprechende Java Runtime Engine. Das Programm bedient sich einiger Java-Bibliotheken, um gewisse Aufgaben ausführen zu können. Hierzu gehören die Libraries *SNMP4J*[2] für die SNMP-Kommunikation, *dnsjava*[3] für DNS-Abfragen (diese Bibliothek arbeitet schneller als die Java-Standardfunktionen dazu) und *Jpcap*[4] (welches einen Java-Adapter zu *WinPcap*[5] darstellt) für den direkten Netzwerkzugriff. Weiters wird *HSQLDB*[6] als Datenbanksystem verwendet. Zur Generierung von Netzwerk-Graphen wird *Graphviz*[7] benutzt, welches jedoch keine Java-Bibliothek ist, sondern ein eigenständiges Programm.

Bei allen genannten Bibliotheken handelt es sich um Open-Source-Software. Diese kann einerseits frei verwendet und andererseits nach Wunsch adaptiert werden. Das war auch in zwei Fällen nötig, nämlich bei der Anpassung von

[1] `http://www.java.com/`
[2] `http://www.snmp4j.org/`
[3] `http://www.xbill.org/dnsjava/`
[4] `http://netresearch.ics.uci.edu/kfujii/jpcap/doc/`
[5] `http://www.winpcap.org/`
[6] `http://hsqldb.org/`
[7] `http://www.graphviz.org/`

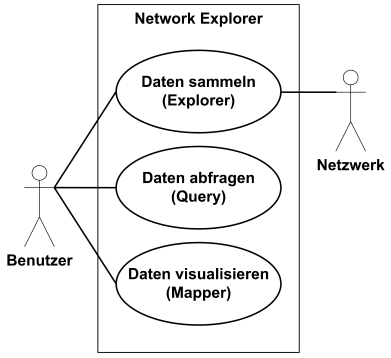


**Abbildung 5.1:** Funktionen des Network Explorers

*Jpcap* (und zwar sowohl von dessen Java- als auch C-Codes), um VLAN getaggte Frames senden und empfangen zu können, sowie dem Hinzufügen einer `GROUP_CONCAT` Funktion zur *HSQLDB*, damit Strings aus mehreren Zeilen zusammengefasst werden können.

Das fertige Programm besteht aus 88 Klassen in 14 Paketen und hat etwa 16.000 Zeilen reinen Quellcode (Kommentare nicht mitgezählt). Weiters existiert für die Software eine vollständige *Javadoc*-Dokumentation. Für die Applikation wurde eine modulare Architektur gewählt, um sie leicht erweiterbar und anpassungsfähig zu machen.

Da das Programm in Java geschrieben wurde und hauptsächlich Java-Bibliotheken verwendet (sogar für die Datenbank), ist es prinzipiell Plattform-unabhängig. Auch *Graphviz*, welches keine Java-Komponente ist, ist für verschiedene Plattformen verfügbar und schränkt die genannte Unabhängigkeit daher nicht ein. Dennoch muss das Programm für bestimmte Tätigkeiten auf spezifische System-Kommandos zurückgreifen (wie z.B. auf `ipconfig` zum Auslesen des Default Gateways). Dadurch kann es in der vorliegenden Version nur auf *Microsoft Windows* eingesetzt werden. Es wurde bei der Entwicklung allerdings beachtet, diese Plattform-spezifischen Aspekte möglichst modular zu gestalten (z.B. die Spezifikation solcher Befehle über Konstanten), damit das Programm leicht auf andere Umgebungen portiert werden kann (für *Unix* müsste z.B. das Kommando `ipconfig` einfach durch `ifconfig` ersetzt werden).

## 5.2 Funktionen

Dieser Abschnitt beschreibt die Funktionen des Programms aus Benutzersicht. *Abbildung 5.1* stellt diese dar. Es wird hier jedoch nicht auf die genaue Bedienung der Applikation eingegangen. Hierzu sei auf Abschnitt *B.4 Bedienung* verwiesen.

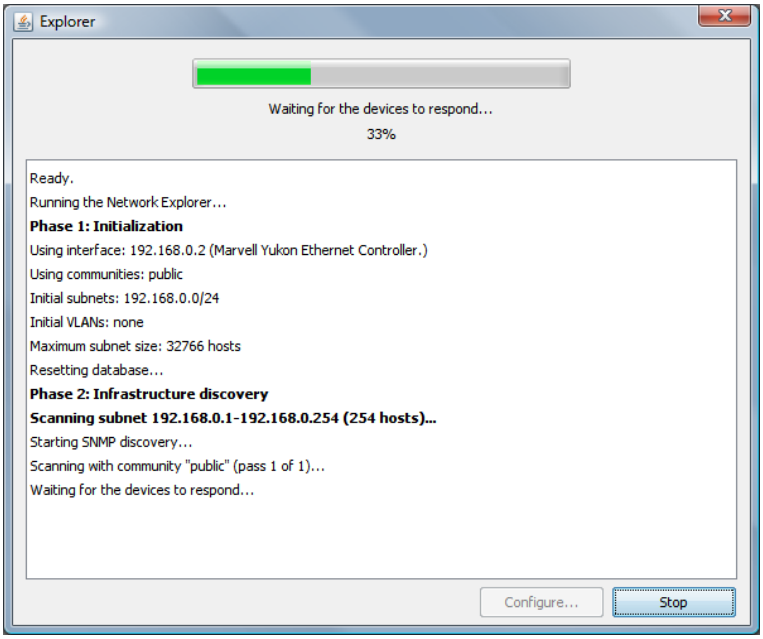


**Abbildung 5.2:** Screenshot der Explorer-Funktion

Kern des Systems ist die **Explorer**-Funktion (siehe *Abbildung 5.2*), welche den zuvor beschriebenen Algorithmus umsetzt und Daten im Netz sammelt. Die Explorer-Funktion initiiert und steuert den Discovery-Prozess. Auf diesen soll hier jedoch nicht weiter darauf eingegangen werden, da der Discovery-Algorithmus bereits zu Genüge betrachtet wurde. Die Architektur der Discovery-Komponenten wird noch separat in Abschnitt *5.3 Architektur* behandelt. Das beim Discovery verwendete Datenmodell wird in Abschnitt *5.4 Datenmodell* beschrieben.

Um die gesammelten Informationen betrachten und auswerten zu können bietet das Programm zwei Funktionen. Die **Query**-Funktion (siehe *Abbildung 5.3*) erlaubt es, die Daten mittels frei definierbaren SQL-Queries abzufragen und tabellarisch darzustellen. Die Daten können dabei beliebig ausgewählt, miteinander verknüpft, gefiltert und sortiert werden, um jeweils genau den gewünschten Aspekt analysieren zu können.

Mit der **Mapper**-Funktion (siehe *Abbildung 5.4*) können aus den gesammelten Daten grafische Netzwerk-Maps erstellt werden, wobei verschiedene Arten (Layer 3 Topologie, Layer 2 Topologie und Spanning Tree) generiert werden können. Zur Berechnung derselben wird auf *Graphviz* zurückgegriffen. Einige von dem Programm erzeugte Maps werden in Abschnitt *6.3 Ergebnisse* präsentiert.

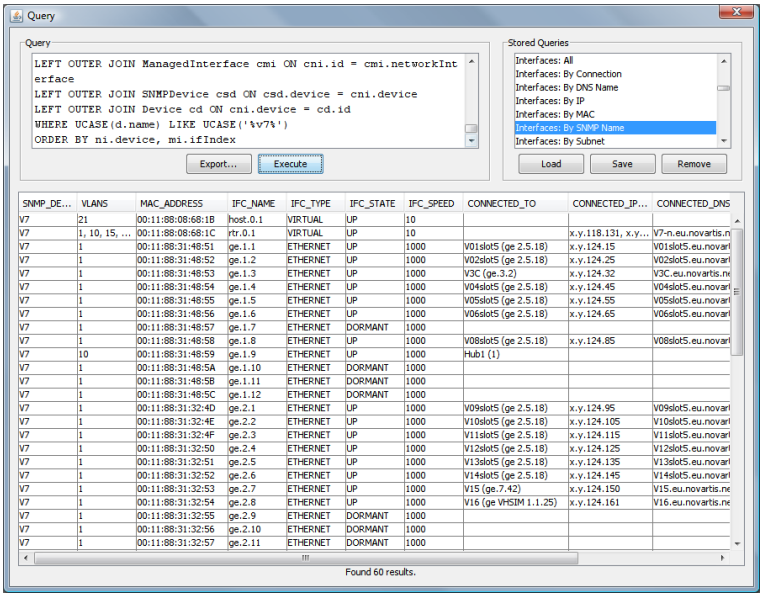


**Abbildung 5.3:** Screenshot der Query-Funktion

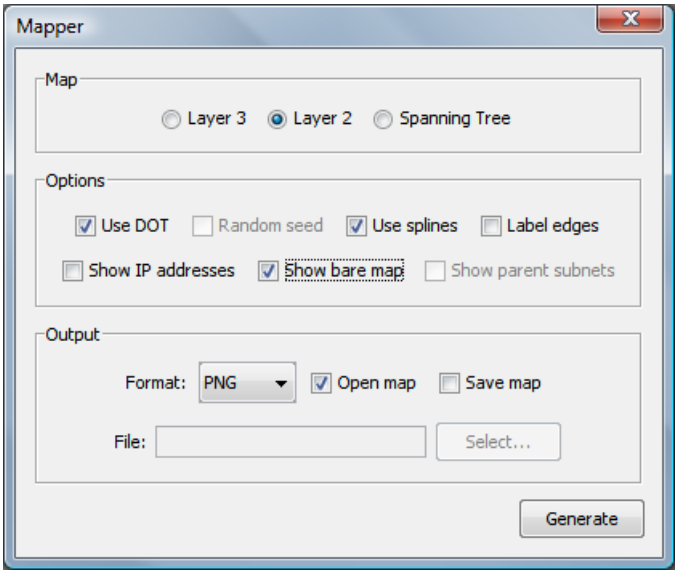


**Abbildung 5.4:** Screenshot der Mapper-Funktion

## 5.3 Architektur

In diesem Abschnitt wird betrachtet, wie das Programm aufgebaut ist. Dabei werden logische, physische und dynamische Aspekte der Architektur beleuchtet. Diese wird hier zwar nicht bis ins letzte Detail beschrieben (dafür sei auf die *Javadoc*-Dokumentation verwiesen), dennoch bietet dieser

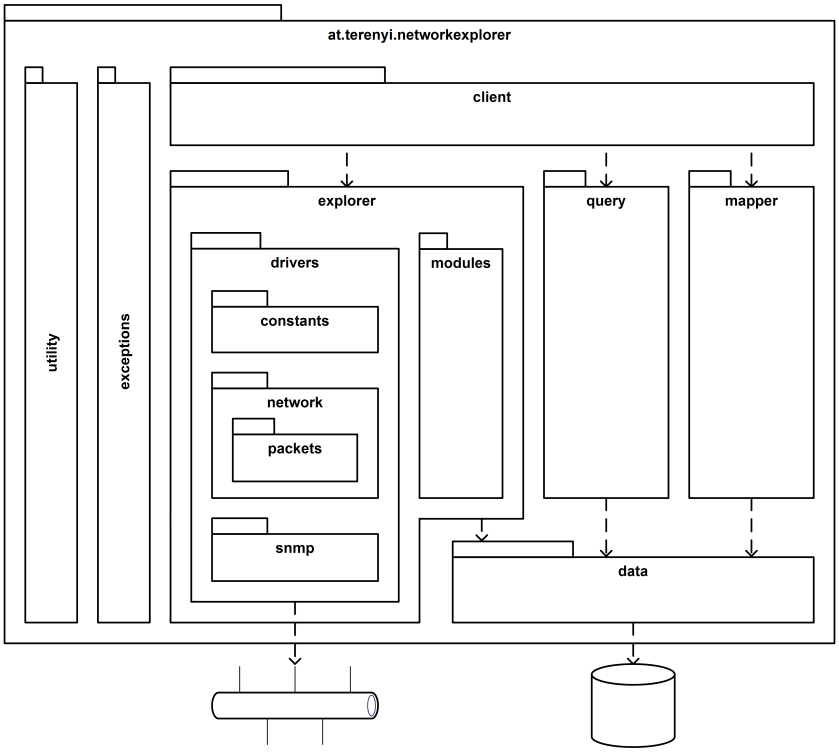


**Abbildung 5.5:** Logische Architektur

Abschnitt eine Orientierung für die Auseinandersetzung mit dem Programm und mit dessen Quellcode. Weiters dient die hier vorgestellte Architektur als Beispiel, wie ein derartiges System aufgebaut werden kann.

### 5.3.1 Logische Architektur

*Abbildung 5.5* zeigt wie die einzelnen Komponenten des Systems logisch gruppiert sind (die Grafik zeigt dabei nur Pakete und nicht einzelne Klassen). Alle Teile des Programms sind im übergeordneten Paket `at.terenyi.networkexplorer` zusammengefasst. Darunter ist die Applikation grob in drei hierarchischen Schichten gegliedert (d.h. eine Ebene bedient sich jeweils der Funktionen der darunterliegenden Ebenen).

Der *Presentation Layer* (das Paket `client`) enthält das User Interface zum Sammeln von Benutzereingaben und zur Darstellung der Ergebnisse. Der *Business Layer* (bestehend aus den Paketen `explorer`, `query` und `mapper`) beinhaltet die eigentliche Geschäftslogik der Funktionen des Programms. Der Presentation Layer bedient sich des Business Layers um das Discovery durchzuführen, Abfragen zu tätigen oder Grafiken zu generieren und stellt dann dessen Resultate dar. Der *Persistence Layer* (das Paket `data`) bildet die Schnittstelle zur Datenbank und abstrahiert den Zugriff auf diese. Über

ihn kann der Business Layer Daten abspeichern und abfragen. Die Pakete `utility` und `exceptions` enthalten überall verwendete Hilfsfunktionen.

Im Folgenden werden die einzelnen Pakete näher betrachtet:

Das Paket `client` enthält das User Interface. Der Programmeinsprungspunkt liegt in der Klasse `NetworkExplorer`, welche gleichzeitig das Hauptfenster definiert. Von dort aus können weitere Windows, welche in den Klassen `ExplorerGUI`, `QueryGUI` und `MapperGUI` implementiert sind, zur Bedienung der einzelnen Funktionen geöffnet werden. Die Explorer-Funktion verwendet ein zusätzliches Fenster (enthalten in der Klasse `SettingsGUI`) zur Eingabe der Konfigurationseinstellungen für das Discovery.

Das Paket `explorer` macht den Kern der Applikation aus. In diesem liegt die Klasse `Explorer`, welche für den Discovery-Prozess zuständig ist. Dieser wird über das `ExplorerGUI` gestartet und dann asynchron in der Klasse `Explorer` ausgeführt (vgl. Abschnitt *5.3.3 Dynamische Architektur*), wobei diese Rückmeldungen über den aktuellen Status an das GUI zurückliefert.

Der eigentliche Discovery-Algorithmus ist wie erwähnt in der Klasse `Explorer` umgesetzt. Die einzelnen Schritte in diesem werden von den Discovery-Modulen übernommen, welche im Unterpaket `modules` enthalten sind und von der `Explorer`-Klasse verwendet und koordiniert werden. Das Interface `Module` definiert eine gemeinsame und einheitliche Schnittstelle für alle Module. Dadurch ergibt sich eine strikt modulare Architektur, die es erlaubt, Module leicht hinzuzufügen bzw. zu ändern und so das gesamte Programm zu erweitern bzw. anzupassen. Natürlich muss bei neuen Modulen der Algorithmus in der `Explorer`-Klasse entsprechend adaptiert werden. Weiters verwenden Module das Interface `ModuleCallback` (das vom Aufrufer implementiert werden muss), um asynchron Meldungen über ihren Status absetzen zu können. In allen Modulen verwendete Funktionen sind in der gemeinsamen Basisklasse `ModuleBase` enthalten.

Zum *Infrastructure Discovery* werden die Module `SNMPDiscovery` (sucht SNMP-fähige Geräte über das Senden von SNMP Requests), `SNMPInformation` (liest von den gefundenen SNMP-Geräten Daten aus und bestimmt damit u.a. vorhandene Subnetze), `SNMPFilter` (identifiziert SNMP-Aliasse, d.h. Geräte die unter verschiedenen IP-Adressen und/oder SNMP Communities die gleichen Informationen liefern) und `VLANDiscovery` (liest die definierten VLANs von allen Switches aus) verwendet.

Zum *Host Discovery* dienen die Module `ARPTableDiscovery` (liest die ARP Tables von Routern aus), `ARPDiscovery` (sendet ARP Requests), `ICMPDiscovery` (sendet ICMP Echo Requests) und `DNSDiscovery` (führt Reverse Name Lookups aus).

Beim *Topology Discovery* werden die Module `STPDiscovery` (bestimmt den Spanning Tree) und `AFTDiscovery` (liest die AFTs der Switches aus) verwendet. Die Inferenz der Topologie sowie die Erkennung von virtuellen Geräten etc. findet in der `Explorer`-Klasse statt. Die einzelnen Module sollen hier nicht näher betrachtet werden, da sie genau die in Kapitel *4 Der Discovery-Algorithmus* beschriebenen Funktionen umsetzen.

Um während des Discoverys über das Netzwerk zu kommunizieren, bedienen sich die Module der Funktionen im Unterpaket `drivers`, welches wiederum einige weitere Unterpakete besitzt. Eines davon ist `constants`, in welchem die Klassen `NetworkConstants` und `SNMPConstants` enthalten sind. Diese definieren gesammelt eine Reihe von Konstanten (z.B. OIDs von SNMP-Variablen).

Das Unterpaket `snmp` stellt Funktionen zur SNMP-Kommunikation bereit. Diese wird über die Klasse `SNMP` abgewickelt, welche intern auf die *SNMP4J*-Bibliothek zurückgreift. Es können SNMP-Anfragen (`Request`) an SNMP-Geräte geschickt werden, welche je eine Antwort (`Response`) liefern. Die Klassen `TableRow` (Zeile einer SNMP-Tabelle) und `Variable` (einzelner SNMP-Wert) stellen die Teile einer derartigen Antwort dar.

Im Unterpaket `network` sind alle Funktionen für den direkten Netzwerkzugriff enthalten. Intern bedienen sich diese Funktionen der *Jpcap*-Bibliothek. Die Klasse `Interface` stellt ein Netzwerk-Interface dar, über welches Pakete gesendet und empfangen werden können. Das Interface `Callback` wird verwendet, damit der Empfang von Daten asynchron behandelt werden kann. Die Hilfsklasse `ARPTable` speichert über ARP gesammelte Informationen. Die Klassen `IPAddress` und `MACAddress` stellen, wie deren Name nahelegt, eine IP- bzw. MAC-Adresse dar. In der Klasse `DNS` sind Funktionen für DNS-Abfragen zusammengefasst, welche auf die *dnsjava*-Bibliothek zurückgreifen.

Die Pakete, welche über ein Interface geschickt bzw. empfangen werden können, sind in dem weiteren Unterpaket `packets` definiert. Alle Pakete haben die gemeinsame Basisklasse `Packet`. Davon leiten sich einige konkrete Pakettypen ab und zwar `ARPPacket`, `IPPacket`, `ICMPPacket`, `TCPPacket` und `UDPPacket`.

Im Paket `query` ist einzig die Klasse `Query` definiert. Diese nimmt eine SQL-Abfrage vom `QueryGUI` entgegen, ruft die entsprechenden Informationen aus der Datenbank ab und gibt diese in Tabellenform zurück.

Das Paket `mapper` enthält die Klasse `Mapper`. Vom `MapperGUI` werden die Optionen des Benutzers an die `Mapper`-Klasse weitergegeben, welche die nötigen Daten aus der Datenbank ausliest und diese für *Graphviz* vorbereitet, welches dann letztendlich die Grafiken erstellt.

Im Paket `data` werden einerseits die Datenobjekte (die alle das Interface `DataObject` implementieren) definiert, welche im Business Layer verwendet werden. Diese werden noch gesondert in Abschnitt *5.4 Datenmodell* im Detail betrachtet. Weiters enthält dieses Paket die Klasse `Database`. Diese dient als Schnittstelle zur Datenbank und erlaubt es Datenobjekte zu speichern, zu ändern, abzufragen und zu löschen (und verwendet dazu die *HSQLDB*-Bibliothek). Dabei kann es mit allen Arten von Datenobjekten umgehen und verwendet Reflection, um automatisch die nötigen SQL-Queries zu generieren. Die Funktionsweise dieser Klasse ist dabei ähnlich zu jener von *Hibernate*. Datenobjekte können daher angepasst werden, ohne dass die Datenbank-Klasse geändert werden müsste.

Das Paket `utility` enthält diverse Hilfsfunktionen. Die Klasse `Formatter` definiert Formatierungsfunktionen, die Klasse `Strings` erlaubt es Text korrekt zu sortieren, auch wenn dieser Zahlen enthält.

Im Paket `exceptions` wird die vom Programm verwendete Exception-Hierarchie definiert. Alle Exceptions haben die gemeinsame Basisklasse `NetworkExplorerException`. Davon leitet eine Reihe von Klassen ab, um verschiedene Arten von Fehlerzuständen zu charakterisieren und gruppieren. Hier sollen jedoch nicht alle Klassen in diesem Paket aufgelistet werden, dazu sei auf die *Javadoc*-Dokumentation verwiesen.

Weiters definieren einige der hier vorgestellten Klassen wiederum innere Klassen (z.B. für Worker Threads oder spezielle Hilfsfunktionen). Auf diese wird hier jedoch ebenfalls nicht näher eingegangen und es sei wiederum auf die *Javadoc*-Dokumentation verwiesen.

### 5.3.2 Physische Architektur

*Abbildung 5.6* zeigt die physische Verteilung der Komponenten. In der derzeitigen Version des Programmes befinden sich alle Teile auf einem einzigen Computer. Durch die modulare (logische) Architektur könnten diese (z.B. User Interface, Discovery-Module und Datenbank) jedoch auch auf mehrere Maschinen aufgeteilt und so beispielsweise Multi-User-Betrieb und ein verteiltes Discovery ermöglicht werden (vgl. Abschnitt *7.2 Erweiterungen*).

Die kompilierten Klassen des Programms sowie aller Java-Bibliotheken sind zwar in einer einzigen JAR-Datei (`Network Explorer.jar`) zusammengefasst, dennoch handelt es sich dabei jeweils um separate Komponenten.

Da es sich um ein Java-Programm handelt, laufen es sowie die verwendeten Bibliotheken in einer Java Virtual Machine (VM). Weil von Java aus kein direkter Zugriff auf das Netzwerk (d.h. senden und empfangen von Ethernet-Frames) möglich ist, wird `Jpcap` benutzt, welches aus zwei Teilen

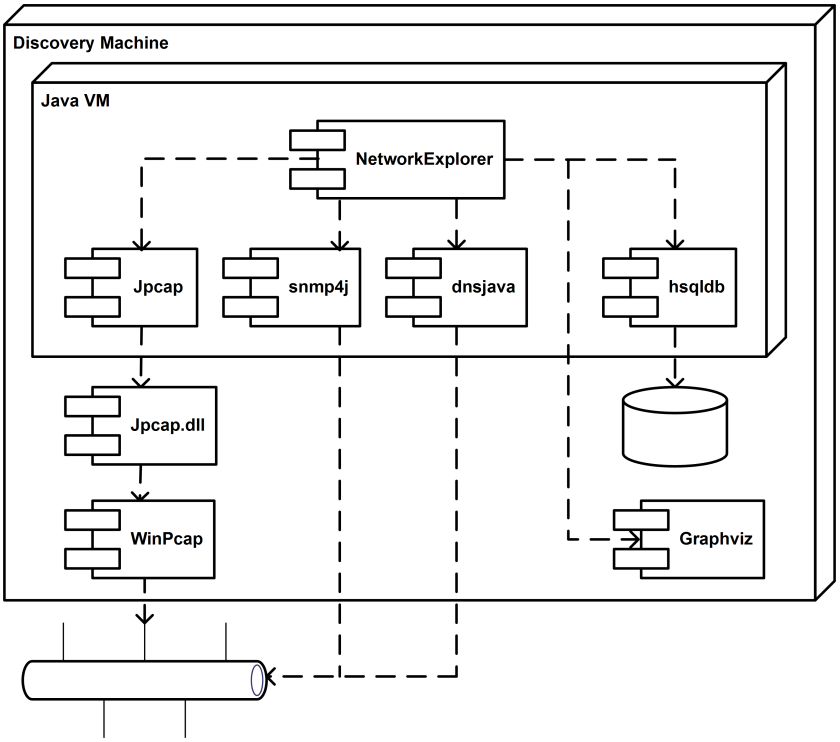


**Abbildung 5.6:** Physische Architektur

besteht. Das Java-Interface greift über JNI auf die in `Jpcap.dll` definierten Funktionen zu (welche in *C* geschrieben und spezifisch für die *Microsoft Windows*-Plattform sind). Diese verwenden wiederum `WinPcap`, welches den direkten Netzwerkzugriff ermöglicht.

Die Bibliotheken `snmp4j` und `dnsjava` implementieren das SNMP- bzw. DNS-Protokoll und werden vom Programm für die entsprechende Kommunikation verwendet. Diese senden und empfangen dazu UDP-Pakete, was direkt von der Java VM unterstützt wird. `Graphviz` ist ein eigenständiges Programm, welches zum Erzeugen von Netzwerk-Graphen aufgerufen wird.

### 5.3.3 Dynamische Architektur

Das Programm verwendet einen Hauptthread, in welchem das User Interface sowie synchron die Query- und Mapper-Funktion laufen. Einzig das Discovery wird asynchron in einem eigenen Thread ausgeführt, damit währenddessen weiterhin Benutzereingaben entgegengenommen werden können (und das Discovery beispielsweise abgebrochen werden kann). Die Explorer-Funktion startet die einzelnen Discovery-Module, welche selbst wiederum mehrere Threads verwenden. Während die Threads der Discovery-Module arbeiten, wartet der Explorer-Thread. Aufgaben jeweils eines Schrit-

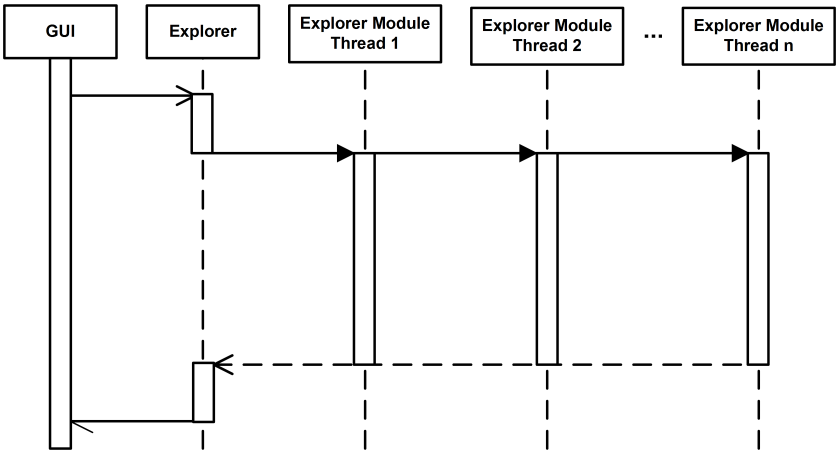


**Abbildung 5.7:** Dynamische Architektur

tes sind zwar parallelisierbar, die einzelnen Schritte müssen jedoch entsprechend dem Algorithmus sequentiell ausgeführt werden. *Abbildung 5.7* stellt dies grafisch dar.

Bei Discovery-Modulen, die Informationen über SNMP abfragen (z.B. ARP Tables oder AFTs auslesen), werden mehrere (und zwar eine vom Benutzer definierbare Anzahl) Threads verwendet, um Daten von mehreren Geräten gleichzeitig einzuholen. SNMP-Operationen sind nicht rechenintensiv und die meiste Zeit wird dabei mit dem Warten auf eine Antwort verbracht. Daher eignen sich diese besonders zur parallelen Verarbeitung – während auf die Antwort von einem Gerät gewartet wird, kann bereits eine Anfrage an ein anderes geschickt werden.

Module, welche Scans durchführen (z.B. ARP- oder ICMP-Pakete verschicken), müssen nicht erst auf Antworten warten, bevor ein neues Paket versendet werden kann, sondern es können gleich alle Pakete auf einmal in kurzer Abfolge abgesetzt werden (lediglich nach dem Versand des letzten Pakets muss eventuell noch ausständigen Antworten etwas Zeit zum Ankommen gegeben werden). Daher ist in diesem Fall eine parallele Ausführung nicht sinnvoll. Dennoch verwenden auch diese Module jeweils zwei Threads: einen zum Versenden der Pakete und einen zweiten, um erhaltene Rückmeldungen asynchron zu verarbeiten.

## 5.4 Datenmodell

Dieser Abschnitt betrachtet das vom Programm verwendete Datenmodell, welches in *Abbildung 5.8* dargestellt ist. Das Modell beschreibt, welche Daten erfasst und persistent in der Datenbank gespeichert werden. Hier soll jedoch nur auf die wichtigsten Zusammenhänge eingegangen werden.

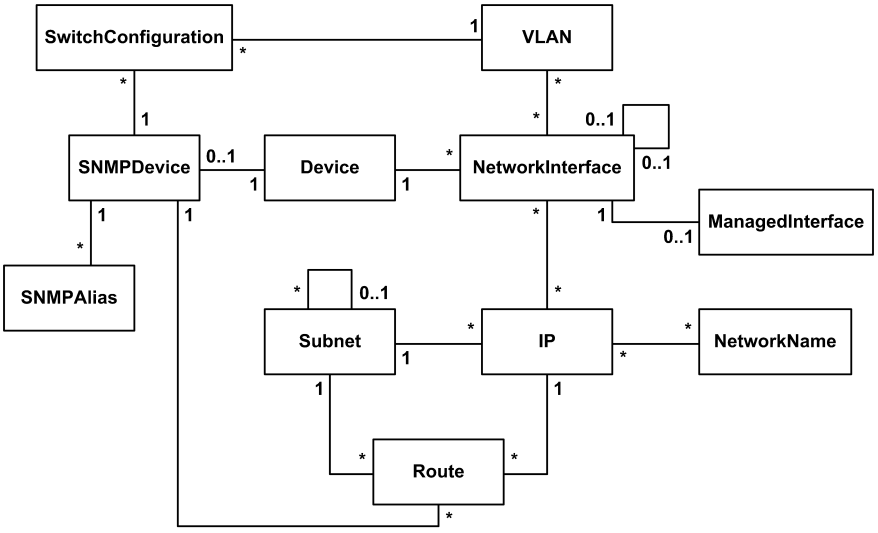


**Abbildung 5.8:** Datenmodell

Für eine detaillierte Beschreibung aller Objekte inklusive ihrer Felder sei auf die *Javadoc*-Dokumentation verwiesen. Das Ziel dieses Abschnitts ist – analog zum Abschnitt *5.3 Architektur* – einerseits als Beispiel zu dienen und andererseits das Verständnis des Programms zu erleichtern.

Im Zentrum des Datenmodells stehen die aufgespürten Geräte (`Device`). Über diese wird deren Typ (z.B. Host, Hub, Router, Switch, Drucker etc.) sowie die Methoden mit der sie gefunden wurden gespeichert. SNMP-Geräte (`SNMPDevice`) stellen eine Erweiterung von normalen Geräten dar, über welche zusätzliche Daten (Management-IP-Adresse, SNMP Community, Name, Beschreibung, Kontaktperson etc.) erfasst werden. Manche SNMP-Geräte können unter verschiedenen IP-Adressen und/oder SNMP Communities erreicht werden. Diese Aliasse (`SNMPAlias`) werden ebenfalls gespeichert.

Jedes Gerät hat ein oder mehrere Interfaces (`NetworkInterface`), welche physisch oder virtuell sein können. Die wichtigste Information über ein Interface ist dessen MAC-Adresse. Über Interfaces von SNMP-Geräten (`ManagedInterface`) werden zusätzliche Daten verwaltet (Bezeichnung, Typ, Status, Geschwindigkeit etc.). Ein Interface kann mit einem anderen Interface verbunden sein (daraus ergibt sich die Layer 2 Topologie).

Weiters werden die im Netzwerk verwendeten VLANs (`VLAN`) erfasst. Einerseits werden diesen Interfaces zugeordnet (daraus ergibt sich welches Gerät sich in welchem VLAN befindet) und andererseits wird erhoben, welches VLAN jeweils mit welchem Namen auf welchem Switch (der ein SNMP-Gerät ist) definiert ist (`SwitchConfiguration`).

Einem Interface können (über dessen MAC-Adresse) IP-Adressen (`IP`) zugeordnet sein, welchen wiederum mit Netzwerknamen (`NetworkName`) assoziiert sein können. Jede IP-Adresse gehört zu einem Subnetz (`Subnet`). Falls überlappende Subnetze definiert sind, kann ein Subnetz auch einem „Parent-Subnetz“ (dem umschließenden Subnetz) zugeordnet werden.

Auf Routern (die ebenfalls SNMP-Geräte sind) können Routen (`Route`) in Zielsubnetze definiert sein, wobei jeweils der nächste Schritt (die IP-Adresse des Next Hops) zu diesem angegeben wird.

# Kapitel 6

# Experimentelle Ergebnisse

Nachdem in den vorigen Kapiteln ein Algorithmus zum Discovery sowie ein System, welches diesen Algorithmus implementiert, vorgestellt wurden, so soll nun dieses System auch in einer realen Umgebung getestet und dessen Praxistauglichkeit bestimmt werden.

Zuerst wird die Umgebung, in der getestet wurde, vorgestellt und dann der genaue Versuchsaufbau, inklusive einer Beschreibung der einzelnen durchgespielten Szenarien. Anschließend werden die vom Programm gelieferten Ergebnisse präsentiert, verglichen und analysiert. Weiters werden besondere Beobachtungen, die im Rahmen der Entwicklung des Programms bzw. der Tests gemacht wurden, gesondert betrachtet. Abschließend wird auf Basis der zuvor vorgestellten Ergebnisse das System als Ganzes bewertet und ein Fazit über die gewonnen Erkenntnisse gezogen.

## 6.1 Versuchsumgebung

Das Programm wurde mit freundlicher Genehmigung im Netzwerk der Firma *Novartis* in Wien Liesing getestet. Dieses Areal war ursprünglich der Standort der *Novartis Institutes for Biomedical Research* (*NIBR*) – die jedoch 2008 größtenteils geschlossen wurden – bzw. ist es immer noch von einigen anderen zum Novartis-Konzern gehörigen Firmen. Da zum Zeitpunkt der Erstellung dieser Arbeit dieser Standort schrittweise stillgelegt wird, befinden sich im Netzwerk im Vergleich zu früher weniger Endgeräte (Workstations, Drucker etc.) – wenn es auch immer noch eine beträchtliche Anzahl ist. Die Infrastruktur (Router, Switches, aber auch Server) wird hingegen noch komplett erhalten, solange bis alle verbleibenden Betriebsteile abgesiedelt sind. Außerdem werden auch andere Standorte von dort aus betreut. Eine derartige Außenstelle befindet sich z.B. in Wels, eine andere an einem anderen Ort in Wien. Wels ist über eine WAN-Verbindung mit dem Areal verbunden und betreibt ein separates kleines LAN (das aber von

Wien aus verwaltet wird), während der andere Standort in Wien netzwerktechnisch Teil des selben LANs ist. Mit dem Rest des weltweiten Novartis-Netzwerkes ist das Gelände ebenfalls über einen weiteren WAN-Link verbunden. Als Testumgebung wurde jedoch nur das LAN in Wien bzw. Wels, welche gemeinsam eine Verwaltungseinheit bilden, verwendet.

Es handelt sich bei dem genannten Standort um ein ideales Testumfeld, da es im Netzwerk einerseits eine Vielzahl von unterschiedlichen Geräten gibt und eine sehr heterogene Umgebung vorliegt – verschiedenste Workstations, Laptops, Drucker, Labor-Spezialgeräte (es handelte sich um ein Forschungsinstitut), *Microsoft Windows*-Server, *Unix*-Server, Server-Cluster, Switches und Router verschiedener Marken und Typen, Firewalls, WAN Router, WLAN Access Points, IP-Telefone, Videokonferenzsysteme, USV-Anlagen usw. Andererseits werden viele der heutzutage typischerweise in einem großen LAN vorkommenden Technologien eingesetzt. Dazu gehören VLANs, das Spanning Tree Protocol, virtuelle Router (und zwar sowohl VRRP als auch HSRP), Link Aggregation (zwischen Servern und Switches sowie zwischen Switches untereinander), ARP Proxies usw. Auch fehlerhafte Einstellungen (z.B. falsche Subnetzmasken) wurden beim Discovery gefunden. Selbst wenn dies aus Sicht der Netzwerkadministration unerwünscht ist, so ist es dennoch beim Testen hilfreich, da geprüft werden kann, wie robust das Programm ist und wie es in solchen Situationen reagiert.

Bei dem genannten Netzwerk wird auf Layer 3 TCP/IP (IPv4) eingesetzt. Auf Layer 2 handelt es um ein Ethernet-Netzwerk (wobei zusätzlich auch noch ein altes FDDI-Netzwerk in Betrieb ist). Man kann sagen, dass das Netzwerk (aus historischen Gründen) aus zwei Hälften besteht, nämlich der des früheren Forschungsinstituts (NIBR) und der der anderen Novartis-Firmen. Diese beiden LANs wurden jahrelang getrennt betrieben und erst vor ca. 5 Jahren miteinander verbunden. Daher stammt die etwas eigentümliche Topologie.

Bei der Infrastruktur auf der NIBR-Seite werden *Enterasys* bzw. *Cabletron* Switches und Router eingesetzt, während auf der anderen Seite *Cisco*-Geräte verwendet werden. Zusätzlich werden auch in einigen Spezialfällen Switches anderer Hersteller (z.B. von *HP*) benutzt. Weiters werden nicht nur Geräte verschiedenster Hersteller, sondern auch unterschiedliche Typen jeweils eines Herstellers verwendet – also eine perfekte Umgebung um zu testen, ob das Programm mit dieser Vielfalt umgehen kann und eine Lösung bietet, die unabhängig von einem bestimmten Hersteller oder Typ funktioniert.

Beide Seiten bestehen konzeptionell aus verschiedenen Ebenen. Den *Core* bilden jeweils zwei Router/Switches auf jeder Seite (wobei die Cisco-Geräte sich als je eine Maschine mit Switching und Routing Funktionen darstellen

und bei den Enterasys-Geräten die Routing Engine separat ansprechbar ist, diese sich also als je zwei Maschinen geben, wobei eine rein virtuell ist). Mit diesem Core verbunden sind einerseits die *Server Switches* (an denen, wie der Name sagt, die Server hängen) und der *Access Layer* (d.h. die Switches, an denen letztendlich alle Endgeräte angeschlossen sind). Zwischen Core und Server Switches sowie zwischen den Server Switches und den Servern wird Link Aggregation eingesetzt. Auf der Cisco-Seite sind die Access Switches U-förmig zusammengeschlossen und an jedem Ende mit einem der beiden Core Switches verbunden (d.h. die Switches sind seriell in einer Kette verkabelt mit den Core Switches an den beiden Enden). Auf der Enterasys-Seite ist jeder Verteiler einzeln mit beiden Core Switches verbunden. Natürlich sind einige dieser Verbindungen durch das STP deaktiviert, um Schleifen zu vermeiden.

Insgesamt gibt es im Netzwerk vier Router, zwei auf jeder Seite, die je zusammen einen virtuellen Router (unter Verwendung von VRRP auf der Enterasys-Seite und HSRP auf der Cisco-Seite) bilden. Weiters gibt es etwa 100 Switches in ungefähr 30 Verteilern (wobei vor allem auf der Enterasys-Seite in vielen – jedoch nicht allen – Fällen jeder Slot eines Verteilers als eigener Switch ansprechbar ist, daher die hohe Anzahl an Switches) und zirka 20 WLAN Access Points. Schließlich gibt es ungefähr noch 500 Endgeräte (deren Zahl aus den zuvor genannten Gründen gesunken ist), wobei die tatsächliche Zahl nur schwierig nachvollzogen werden kann, weil diese Geräte auch oft aus- und eingeschaltet werden. Es war daher nicht nachprüfbar, wie viele Geräte zu einem bestimmten Zeitpunkt im Netz hätten aktiv sein müssen. Ansonsten sind die Geräte über etwa 25 verschieden Subnetze verteilt und es sind 33 VLANs definiert. Insgesamt umfasst der verwendete IP-Bereich etwa 3.800 Adressen.

Zum Überprüfen der vom Programm gelieferten Daten boten sich zwei Quellen an, die einander ideal ergänzen. Einerseits gibt es eine schriftliche Netzwerkdokumentation, in der alle Fakten über die Infrastruktur inklusive deren Topologie, jedoch aber keine Daten über Endgeräte verzeichnet sind. Andererseits ist ein dem hier vorgestellten Programm ähnliches System, genannt *Network Client Locator* (*NCL*), im Einsatz, welches automatisch nach Endgeräten sucht und deren Standort bestimmt. Dafür wird jedoch die Topologie von diesem System nicht erfasst und die Daten über die Infrastruktur müssen zuerst händisch eingegeben werden.

## 6.2 Versuchsaufbau

Das Discovery wurde von einem eigens dafür eingerichteten PC (einer physischen Maschine) aus durchgeführt. Dieser ist an einem gewöhnlichen Access

| Datum | Szen. 1 | Szen. 2 | Szen. 3 | Szen. 4 |
|---|---|---|---|---|
| Gültige Communities | 8 | 8 | 0 | 0 |
| Definierte Subnetze | 0 | 0 | 26 | 26 |
| Definierte VLANs | 0 | 0 | 0 | 33 |
| Inter-Packet Delay | 1ms | 1ms | 1ms | 1ms |
| Packets per IP | 1 | 2 | 1 | 1 |
| SNMP Timeout | 3000ms | 3000ms | 3000ms | 3000ms |
| SNMP Retry Count | 1 | 1 | 1 | 1 |
| Threads | 50 | 50 | 50 | 50 |
| Table Size | 5000 | 5000 | 5000 | 5000 |
| Subnet Size Limit | 32766 | 32766 | 32766 | 32766 |

**Tabelle 6.1:** Eingabedaten und Einstellungen

Switch angeschlossen, befindet sich also an keiner besonderen Stelle im Netzwerk. Einzig der Switch Port, an dem die Discovery-Maschine angeschlossen ist, wurde so konfiguriert, dass über ihn VLAN-getaggte Ethernet-Frames in alle VLANs gesendet und aus allen VLANs empfangen werden können.

Insgesamt wurden vier verschiedene Szenarien getestet. In den Szenarien 1 und 2 waren die einzigen Eingabedaten lediglich die verwendeten SNMP Communities (insgesamt 8 an der Zahl), damit das Programm Switches und Router finden und alle weiteren Infrastruktur-Daten (Subnetze, VLANs) von diesen auslesen kann. Szenario 2 unterscheidet sich von Szenario 1 nur dahingehend, dass bei jeder Scan-Operation jeweils zwei, anstatt nur einem Paket an jede IP-Adresse gesandt wurde, in dem Gedanken, dass dies die Ergebnisse verbessern könnte (jedoch auf Kosten der Durchlaufzeit), falls etwa Pakete verlorengehen.

In den Szenarien 3 und 4 wurde getestet, wie sich das Programm verhält, wenn keine Daten über SNMP verfügbar sind. In diesen wurde daher nur je eine ungültige SNMP Community (das Programm verlangt, dass mindestens eine Community angegeben wird) verwendet, worauf aber keine Geräte antworten sollten und so keine Informationen über SNMP bezogen werden können. Da nun das Programm aber die Subnetze (außer dem initialen, in dem sich die Discovery-Maschine befindet) nicht mehr bestimmen kann, müssen diese explizit angegeben werden. Bei Szenario 4 wurden, zusätzlich zu den zu untersuchenden Subnetzen, auch noch die VLANs definiert.

*Tabelle 6.1* listet die verwendeten Eingabedaten und Einstellungen (vgl. Abschnitt *B.4.1 Explorer*) noch einmal auf. Das Subnet Size Limit wurde auf 32.766 gesetzt, um sicherzustellen, dass nur zum lokalen LAN gehörige Subnetze gescannt werden (diese sind alle kleiner als die genannte Größe).

| **Methode** | **Szen. 1** | **Szen. 2** | **Szen. 3** | **Szen. 4** |
|---|---|---|---|---|
| *Infrastructure Discovery* | *04:50* | *06:13* | *01:25* | *01:25* |
| ARP Table Discovery | 02:56 | 03:51 | 00:00 | 00:00 |
| ARP Discovery | 05:07 | 09:14 | 00:09 | 04:54 |
| ICMP Discovery | 00:14 | 00:25 | 00:10 | 00:10 |
| DNS Discovery | 00:29 | 00:33 | 00:25 | 00:26 |
| *Host Discovery* | *08:49* | *14:07* | *00:46* | *05:32* |
| STP Discovery | 00:08 | 00:10 | 00:00 | 00:00 |
| AFT Discovery | 01:26 | 01:43 | 00:15 | 00:16 |
| Topology Inference | 00:12 | 00:12 | 00:00 | 00:00 |
| *Topology Discovery* | *01:48* | *02:06* | *00:15* | *00:17* |
| **Gesamt** | **15:39** | **22:38** | **02:32** | **07:20** |

**Tabelle 6.2:** Dauer des Discoverys

Alle Tests wurden am Nachmittag eines normalen Werktages durchgeführt, damit während der Tests möglichst viele Geräte eingeschaltet und mit dem Netzwerk verbunden sind. Die Tests wurden jedoch hintereinander und nicht parallel ausgeführt, wodurch es möglich ist, dass bei den einzelnen Szenarien unterschiedlich viele Endgeräte aktiv waren.

## 6.3 Ergebnisse

In diesem Abschnitt sollen nun die Ergebnisse der einzelnen Test dargestellt, verglichen, kommentiert und analysiert werden.

*Tabelle 6.2* zeigt, wie viel Zeit für das Discovery in den einzelnen Fällen vonnöten war. Erwartungsgemäß hat Szenario 2 am längsten gedauert. Darauf folgt Szenario 1, da hier bei den Scans nur jeweils ein Paket an jede IP-Adresse gesendet wurde. Nur etwa halb so lange dauerte Szenario 4, da hier keine Daten über SNMP ausgelesen werden mussten bzw. konnten. Daran zeigt sich, dass dieses Auslesen von Informationen über SNMP einen nicht unwesentlichen Anteil an der Gesamtdauer ausmacht. Am kürzesten dauerte Szenario 3, da hier zusätzlich beim ARP Discovery nur ein Mal der Adressbereich und nicht alle VLANs gescannt wurden. Es ist zu sehen, dass das ARP Discovery der am längsten dauernde Arbeitsschritt ist.

*Tabelle 6.3* zeigt die Anzahl der gefundenen Geräte, aufgeschlüsselt nach deren Typ. *Tabelle 6.4* zeigt mit welcher Methode diese Geräte gefunden wurden.

| Typ | Szen. 1 | Szen. 2 | Szen. 3 | Szen. 4 |
|---|---|---|---|---|
| Hosts | 474 | 464 | 622 | 643 |
| SNMP | 304 | 308 | 2 | 2 |
| Hubs | 19 | 19 | 0 | 0 |
| Router | 13 | 13 | 0 | 0 |
| Switches | 103 | 103 | 0 | 0 |
| WLAN APs | 17 | 17 | 0 | 0 |
| Drucker | 67 | 68 | 0 | 0 |
| Andere SNMP | 124 | 127 | 2 | 2 |
| WAN Gateways | 2 | 2 | 0 | 0 |
| ARP Proxies | 1 | 1 | 0 | 0 |
| VRRP/HSRP | 3 | 3 | 1 | 3 |
| VMware | 32 | 32 | 6 | 20 |
| **Gesamt**[1] | **2049** | **2047** | **2041** | **2051** |

**Tabelle 6.3:** Anzahl gefundener Geräte

| Methode | Szen. 1 | Szen. 2 | Szen. 3 | Szen. 4 |
|---|---|---|---|---|
| SNMP Discovery | 304 | 308 | 2 | 2 |
| ICMP Discovery | 576 | 549 | 624 | 604 |
| ARP Discovery | 569 | 555 | 104 | 574 |
| ARP Table Discovery | 678 | 673 | 0 | 0 |
| AFT Discovery | 704 | 682 | 0 | 0 |
| DNS Discovery | 1940 | 1940 | 2041 | 2015 |
| Route Discovery | 14 | 14 | 0 | 0 |
| **Gesamt**[2] | **2030** | **2028** | **2041** | **2051** |

**Tabelle 6.4:** Gefundene Geräte pro Methode

Auffällig ist, dass für viele Geräte ein Eintrag im DNS existiert, diese aber zum Zeitpunkt des Discoverys nicht aktiv waren. Dadurch wurden die meisten Geräte im DNS Discovery entdeckt, die tatsächlich (mit den anderen Methoden) im Netz gefundenen Geräte machen einen Bruchteil davon aus.

Die unterschiedliche Anzahl an gefundenen Geräten im DNS Discovery zwischen den Szenarien 1 und 2 einerseits sowie 3 und 4 andererseits ergibt sich dadurch, dass bei den Szenarien 3 und 4 auf Grund der fehlenden SNMP-Informationen zusammengehörige Interfaces nicht erkannt werden konnten

[1] Die Gesamtzahl ist nicht die Summe der einzelnen Zeilen in einer Spalte, da ein Gerät mehrere Funktionen haben kann (z.B. Switch und WLAN Access Point).

[2] Auch hier können die Zeilen nicht einfach zur Gesamtzahl aufsummiert werden, da ein Gerät mit mehreren Methoden gefunden werden kann.

und daher scheinbar mehr eigenständige Geräte gefunden wurden (siehe auch weiter unten). Der Unterschied in der Anzahl zwischen dem Szenario 3 und 4 rührt analog dazu daher, dass bei Szenario 4 (durch das ARP Discovery in allen VLANs) mehr MAC-Adressen mit den IP-Adressen assoziiert und über diese Zuordnung zusammengehörige Daten zusammengefasst werden konnten, wodurch sich weniger eigenständige Geräte ergeben.

Die Gesamtzahl der gefundenen Geräte ist sowohl zwischen den Szenarien, als auch zwischen den einzelnen Methoden relativ konsistent. Die größte Diskrepanz liegt bei der Anzahl entdeckten Geräten beim ARP Discovery bei Szenario 3, dies liegt jedoch daran, dass bei allen anderen Szenarien alle VLANs gescannt wurden, anstatt nur das VLAN, in dem sich die Discovery-Maschine befindet (da nur Frames ohne VLAN Tag gesendet wurden) und daher entsprechend mehr gefunden wurde.

Bei den Infrastrukturkomponenten wurde überall (wo möglich, d.h. in den Szenarien 1 und 2) die exakt gleiche Anzahl gefunden. Die hohe Anzahl von Switches ist, wie erwähnt, darauf zurückzuführen, dass bei vielen Enterasys-Switches jeder Slot eines Verteilers als eigener Switch ansprechbar ist und entsprechend separat erkannt wird.

Einzig bei den Endgeräten gab es leichte Abweichungen in der Anzahl, was aber höchstwahrscheinlich darauf zurückzuführen ist, dass einerseits zwischen den Testläufen Geräte ein- oder ausgeschaltet wurden bzw. es andererseits auch andere Faktoren gibt, die das Ergebnis nicht-deterministisch machen (z.B. wenn ein Gerät zu spät oder gar nicht antwortet, weil es gerade unter Last steht oder ein Eintrag in einer Tabelle zu einem ungünstigen Zeitpunkt entfernt wird usw.) – dies ist jedoch Teil des Algorithmus, in vollem Bewusstsein, dass ein hundertprozentig vollständiges Ergebnis nicht garantiert (und auch nicht nachgeprüft) werden kann.

Weiters fällt auf, dass die höhere Anzahl von Paketen pro IP-Adresse in den Scans bei Szenario 2 das Ergebnis nicht verbessert hat. Es ist daher davon auszugehen, dass keine Pakete im Netzwerk verloren gingen und eher andere Faktoren (z.B. Antwortzeit der Geräte) Auswirkungen auf die Qualität des Ergebnisses haben.

Bei den Szenarien 3 und 4 konnten die Geräte nicht klassifiziert werden, da die SNMP-Informationen fehlten. Dass in diesen beiden Szenarien trotz der Verwendung einer ungültigen SNMP Community zwei SNMP-Geräte gefunden wurden, liegt daran, dass es zwei SNMP-Geräte gibt, die auf jede beliebige SNMP Community antworten. Dass bei diesen Szenarien weiters bei manchen Methoden bzw. insgesamt auf den ersten Blick mehr Geräte gefunden wurden, als bei den Szenarien 1 und 2, liegt wie erwähnt daran, dass

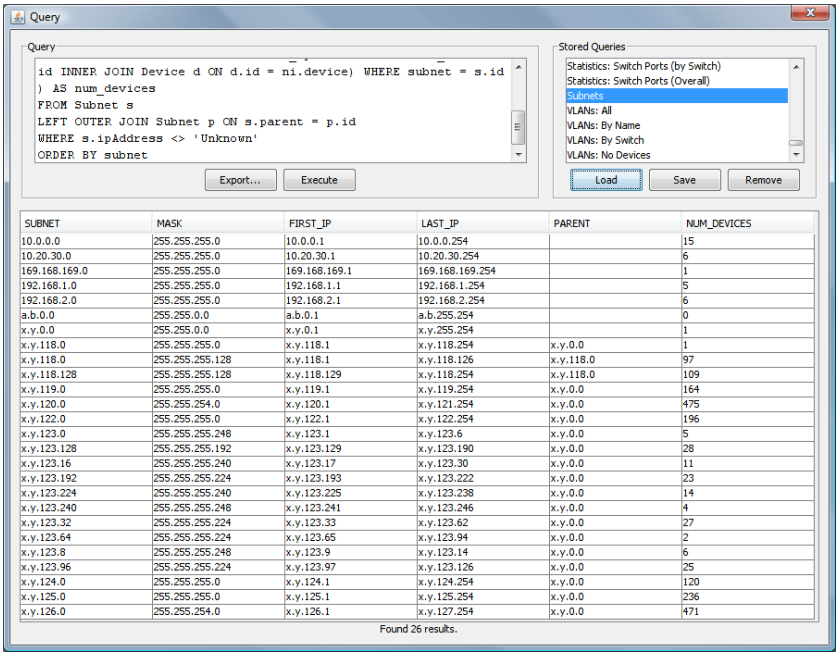


**Abbildung 6.1:** Gefundene Subnetze

nicht tatsächlich mehr Geräte gefunden wurden, sondern dies nur so scheint, da auf Grund der fehlenden SNMP-Informationen zusammengehörige Interfaces nicht erkannt wurden und diese so als eigenständige Geräte geführt werden.

Ein anderer Grund, warum eine scheinbar höhere Anzahl an Geräten gefunden worden sein kann, ist, falls es zu SNMP Timeouts bei Geräten kam, die auf mehrere IP-Adressen und/oder Communities antworten. Diese werden in einem ersten Schritt als (pro IP-Adresse bzw. Community) separate Geräte erkannt und dann (anhand der über SNMP ausgelesenen Daten) zusammengefasst. Kam es aber bei einem Versuch mit einer IP-Adresse bzw. Community zu einem Timeout, so sind in diesem Fall die Daten nicht vollständig und können den anderen nicht zugeordnet werden. Dadurch bleibt ein Alias als eigenes Gerät bestehen und erhöht so die Gesamtzahl.

Neben der absoluten Anzahl der gefundenen Ergebnisse, sind weitere Fragen, wie korrekt und wie vollständig (im Rahmen des Möglichen) diese Ergebnisse sind. Der Abgleich der Ergebnisse mit den Daten der schriftlichen Netzwerkdokumentation sowie mit jenen des NCL-Systems zeigt, dass der Network Explorer sowohl die Infrastruktur (inklusive deren Topologie), als auch die Endgeräte (soweit es nachprüfbar ist) vollständig und korrekt erkannt hat.

*Abbildung 6.1* zeigt eine Liste der gefundenen Subnetze (offizielle, also nicht-private IP-Adressen werden hier zensiert dargestellt), *Abbildung 6.2* eine der

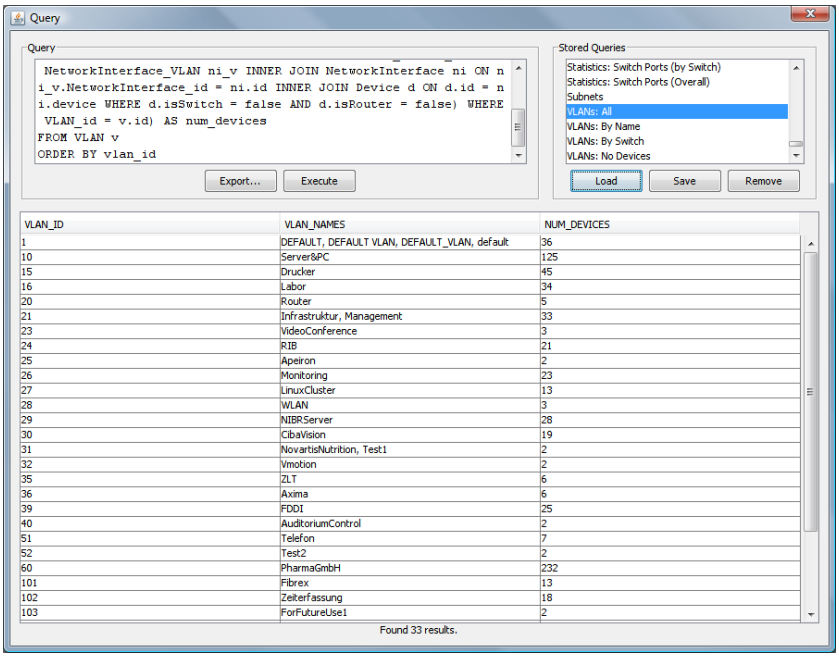


**Abbildung 6.2:** Gefundene VLANs

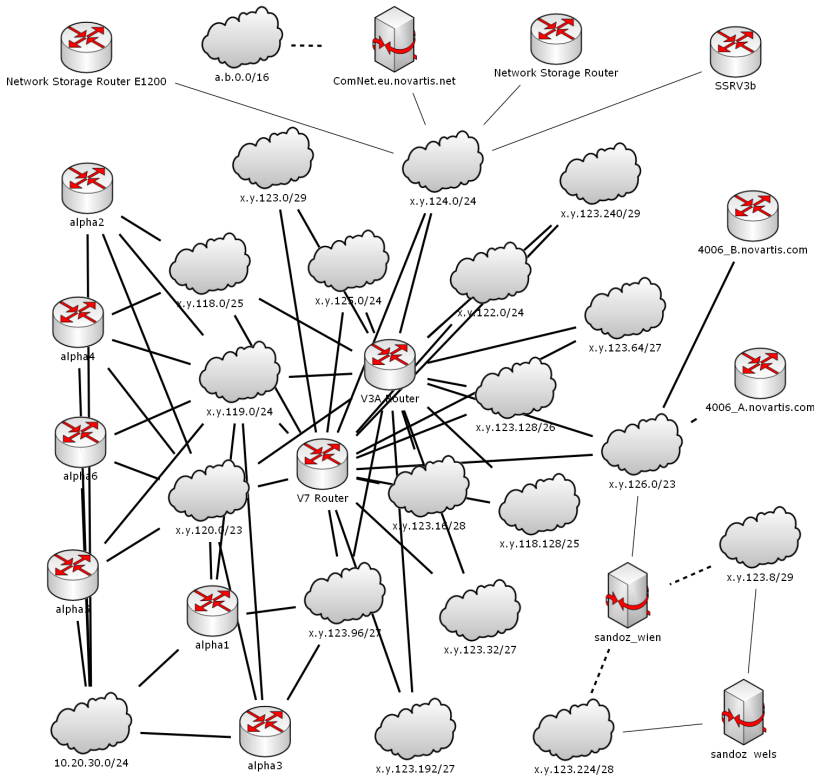


**Abbildung 6.3:** Automatisch bestimmte Layer 3 Map

gefundenen VLANs. Beide Listen sind korrekt und vollständig bzw. wurden einige zusätzliche Subnetze erkannt, welche nicht in der Dokumentation zu finden sind, weil sie von bestimmten Geräten in Sonderfällen verwendet werden (z.B. private Subnetze in einem Cluster).

*Abbildung 6.3* zeigt die Layer 3 Struktur des Netzwerkes (dünne Linien sind Interfaces, dicke Linien sind direkte Routen, strichlierte Linien sind indirekte Routen). Diese wurde ebenfalls korrekt erkannt. `V3A Router` und `V7 Router` sind die Router im Enterasys-Core (wobei die Geräte auf dieser Seite über viele verschiedene Subnetze verteilt sind), `4006_A` und `4006_B` die des Cisco-Cores (wobei die Geräte auf dieser Seite sich alle im gleichen Subnetz `x.y.126.0/23` befinden).

Ebenfalls korrekt erkannt wurde der WAN-Link nach Wels mit den Gateways auf beiden Seiten (`sandoz_wien` und `sandoz_wels`). Weiters wurde `sandoz_wien` richtigerweise als ARP Proxy markiert.

Die Geräte im 124er-Subnetz (z.B. `Network Storage Router`) sind diverse Spezialgeräte, z.B. für ein *Storage Area Network* (*SAN*). Sie werden deswegen alle in diesem Subnetz gefunden, da sich in diesem die Management Interfaces aller Infrastrukturkomponenten befinden.

Die Geräte `alpha1`, `alpha2` etc. sind die einzelnen Maschinen eines Alpha-Clusters. Diese erfüllen zwar keine Routing-Funktionen im Netzwerk, könnten dies aber und gegeben sich daher über SNMP als Router zu erkennen. Das Subnetz `10.20.30.0/24` dient der internen Kommunikation der Cluster-Nodes.

*Abbildung 6.4* zeigt den Cisco-Core und die Server Switches dieser Seite. *Abbildung 6.5* zeigt die Server Switches der Enterasys Seite. Beide sind Ausschnitte aus der Layer 2 Topologie des Netzwerkes (die gesamte Map ist zu groß, um hier dargestellt werden zu können). Strichlierte Linien sind über STP deaktivierte Links. Weiters sind die mehrfachen Verbindungen von Geräten mit einem und/oder mehreren Switches (Link Aggregation bzw. mehrfacher Anschluss zur Ausfallssicherheit) gut zu sehen.

Neben einigen Hubs, die so auch tatsächlich existieren, wurde auch viele Hubs vom Programm erkannt, die nicht physisch vorhanden sind. Dies kommt vor allem bei virtuellen Maschinen bzw. ähnlichen Fällen (z.B. `Hub17`) vor, wo ein Gerät über ein Interface mehrere MAC-Adressen verwendet, falls diese Interfaces nicht über SNMP einem einzelnen Gerät zugeordnet werden können. In diesem Fall gelten alle MAC-Adressen als eigene Interfaces von separaten Geräten bzw. die VMs als eigene Maschinen (obwohl diese nur auf einer physischen Maschine existieren), die alle am selben Switch Port hängen und somit über einen Hub verbunden sein müssen. Ein anderer Fall

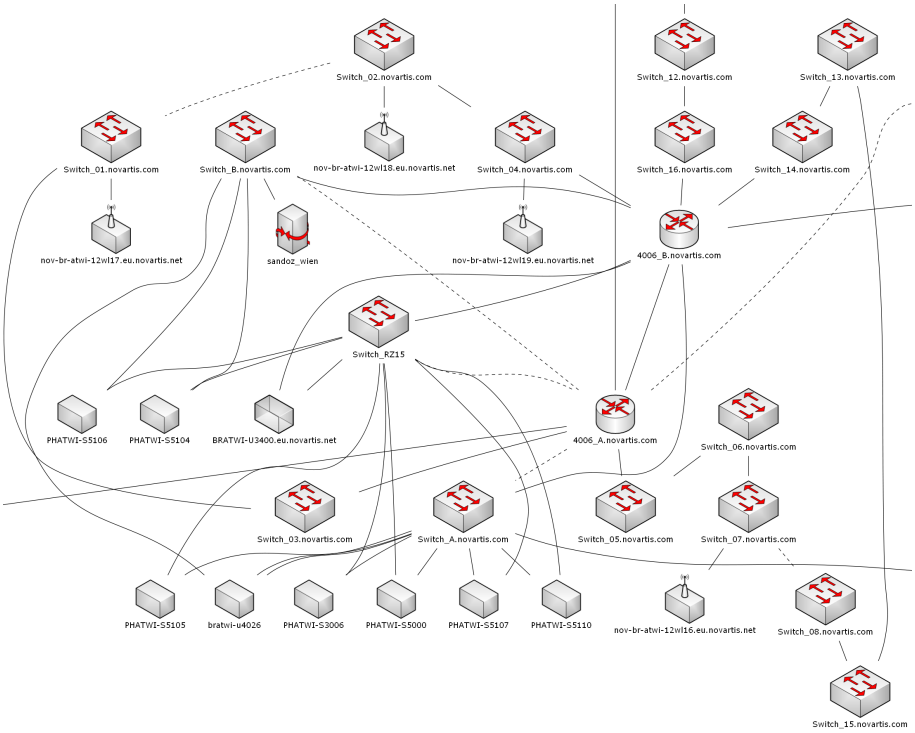


**Abbildung 6.4:** Ausschnitt aus der Layer 2 Map (1)

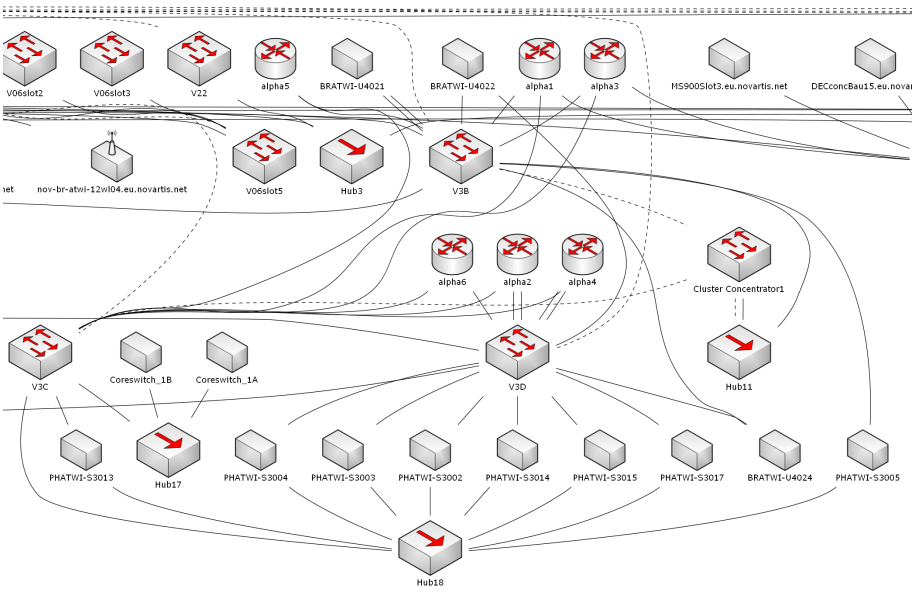


**Abbildung 6.5:** Ausschnitt aus der Layer 2 Map (2)

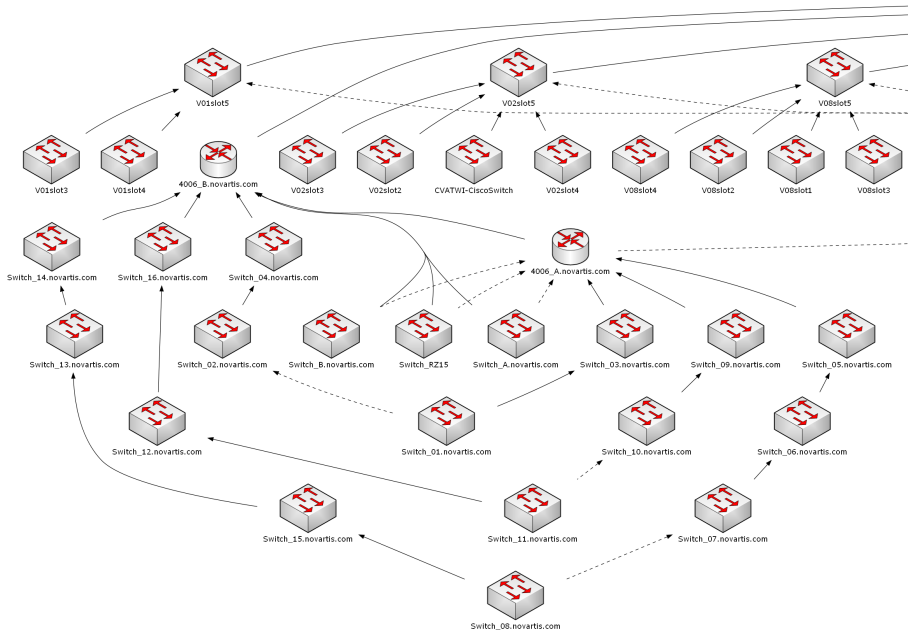


**Abbildung 6.6:** Ausschnitt aus dem Spanning Tree

(z.B. `Hub11`) ist, wenn Link Aggregation über SNMP nur einseitig gemeldet wird. So gibt z.B. der `Cluster Concentrator1` an, dass zwei seiner physischen Ports (anstatt einem virtuellen Link Aggregation Port) mit einem Port (dem virtuellen Link Aggregation Port) von `V3B` (anstatt mit zwei von dessen physischen Ports) verbunden sind. Dadurch, dass nun zwei Ports auf der einen Seite mit einem Port auf der anderen Seite verbunden sind, wird ein Hub dazwischen generiert. Insofern entspricht die gelieferte Information zwar nicht hundertprozentig der Realität, es wurde jedoch das korrekt bestimmt, was anhand der verwendeten und verfügbaren Daten bestimmt werden konnte.

*Abbildung 6.6* zeigt einen Ausschnitt aus dem Spanning Tree. Diese Abbildung unterscheidet sich von der Layer 2 Map dadurch, dass nur Switches und deren durch STP definierte Hierarchie dargestellt werden.

*Abbildung 6.7* zeigt eine Liste von gefundenen Geräten, inklusive einiger Informationen zu ihnen (z.B. IP-Adresse, Netzwerkname, VLAN, Anschlussort etc.). *Abbildung 6.8* zeigt eine statistische Übersicht über alle Switches und deren Ports (z.B. wie viele Ports auf welchem Switch frei oder belegt sind). *Abbildung 6.9* zeigt die Liste der Interfaces eines bestimmten Switches (inkl. Bezeichnung, Status, Geschwindigkeit, definiertes VLAN, angeschlossene Geräte etc.).

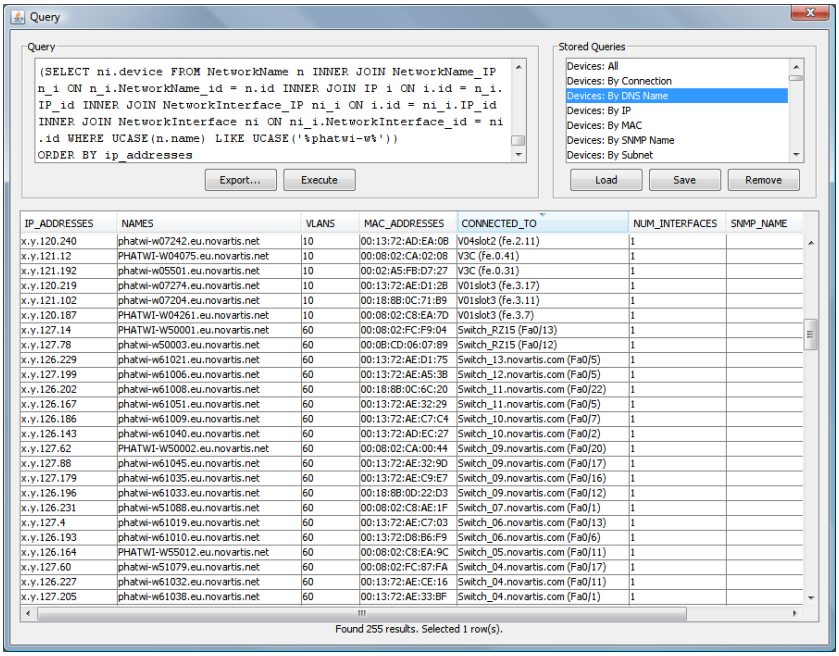


**Abbildung 6.7:** Liste von Geräten

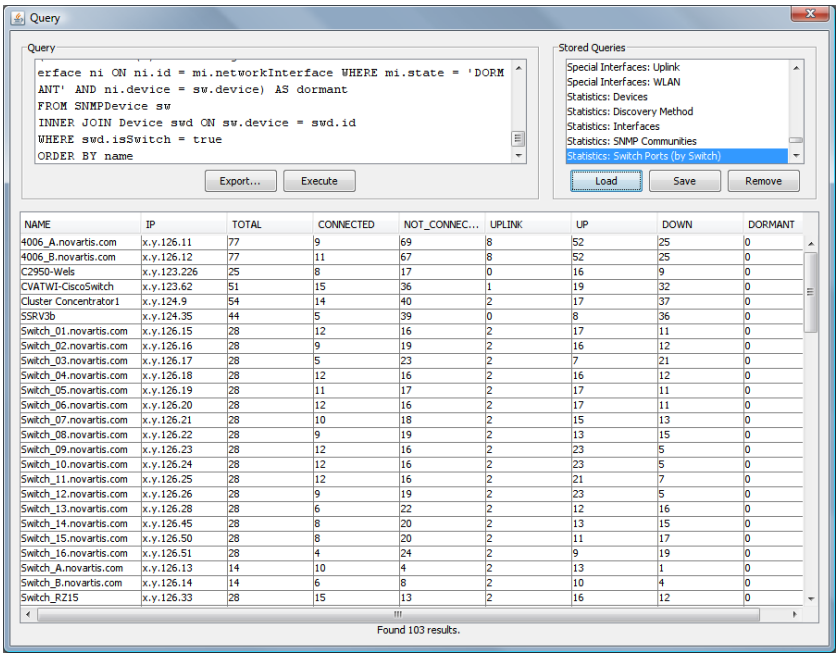


**Abbildung 6.8:** Switch Port Statistiken

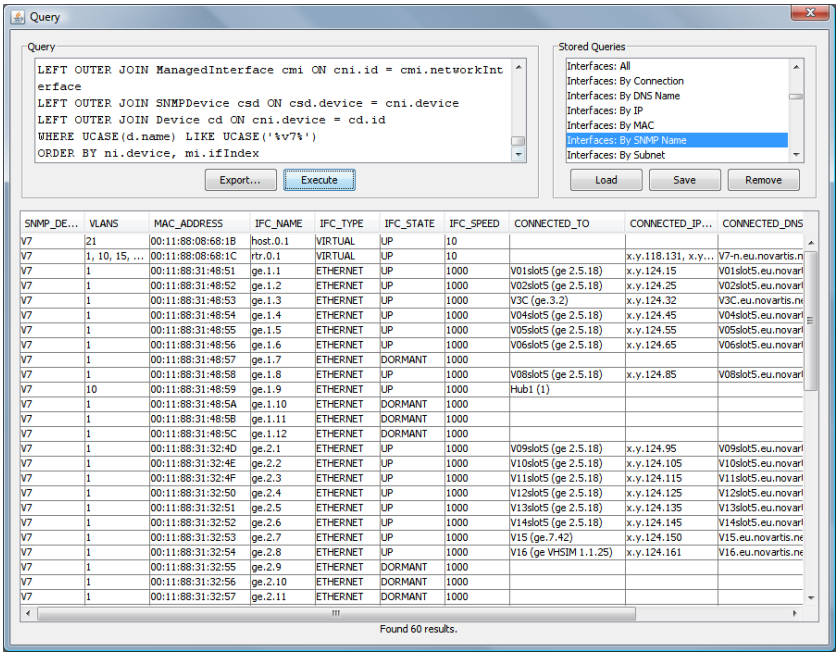


**Abbildung 6.9:** Interfaces eines Switches

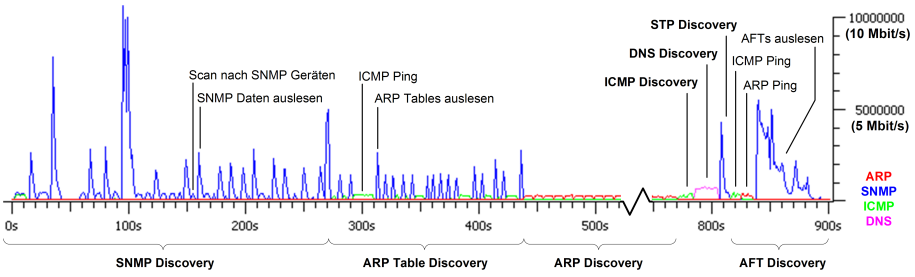


**Abbildung 6.10:** Netzwerklast beim Discovery

Eine weitere Frage, neben Vollständigkeit und Korrektheit der gelieferten Informationen, ist der Ressourcenverbrauch des Programms, wobei hier besonders Speicherverbrauch und Netzwerklast betrachtet werden.

Der maximale Speicherverbrauch lag (bei Szenario 1) bei ca. 260 MB und sollte somit für moderne Computer kein Problem darstellen. *Abbildung 6.10* zeigt die Netzwerklast (bestimmt mit *Wireshark*) während des Discoverys (ebenfalls bei Szenario 1).

Zur Verdeutlichung zeigt *Abbildung 6.11* die Netzwerklast beim Kopieren einer 500 MB großen Datei über das Netzwerk. In diesem Maßstab wäre die Belastung durch das Discovery gar nicht sichtbar.

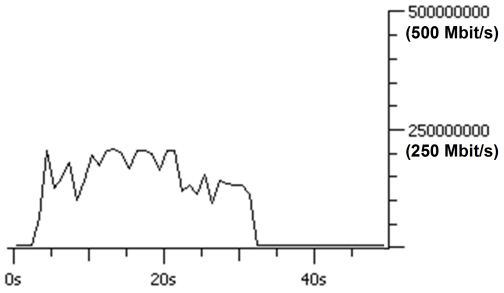


**Abbildung 6.11:** Netzwerklast bei einer Dateiübertragung

Wie zu sehen ist, liegt beim Discovery die Netzwerklast über weite Strecken unter 1 Mbit/s (das ist ein Tausendstel einer Gigabit Ethernet Verbindung). In nur sechs Fällen gibt es einen kurzen Spike über 5 Mbit/s, davon in einem knapp über 10 Mbit/s. Weiters treten diese Spikes nicht beim Scannen, sondern beim Auslesen von Daten über SNMP aus. Dabei wird die Geschwindigkeit vom Kommunikationspartner (z.B. wie schnell ein Switch Antworten schickt) und nicht vom Programm selbst bestimmt.

Im Vergleich dazu ist die Last bei der Dateiübertragung mit bis zu 250 Mbit/s etwa 50 Mal höher (die Discovery-Maschine besitzt ein Gigabit Interface und hängt an einem Gigabit Port des Switches). Daher kann gesagt werden, dass der vom System erzeugte Datenverkehr nicht einmal merklich, geschweige denn für ein modernes Netzwerk belastend ist.

Weiters sind in der *Abbildung 6.10* die einzelnen Schritte des Discoverys klar zu erkennen (die Linien sind je nach Protokoll eingefärbt). Während des SNMP Discoverys machen die kleinen Hügel den Scan nach SNMP-Geräten aus, während die Spitzen das Auslesen der Daten von den gefundenen Geräten darstellen. Beim ARP Table bzw. AFT Discovery ist zu sehen, wie zuerst die Geräte (z.B. im jeweiligen Subnetz beim ARP Table Discovery) mit ICMP und ARP (nur beim AFT Discovery) gepingt und danach über SNMP die entsprechenden Tabellen ausgelesen werden.

## 6.4 Besondere Beobachtungen

In diesem Kapitel werden einige Beobachtungen angeführt, die während der Tests gemacht wurden und einer besonderen Erwähnung wert sind (in keiner bestimmten Reihenfolge). Vieles davon ist auf die Heterogenität der untersuchten Netzwerkumgebung zurückzuführen.

- Dass mehr als die vier Core-Router gefunden wurden, liegt daran, dass einige Geräte Routing-Funktionen besitzen und sich daher auch als Router zu erkennen geben, auch wenn deren Funktionalität deakti-

viert ist und das Routing nur von den vier Core-Routern übernommen wird. Welcher Router wofür Routing-Informationen hat, wird durch das Auslesen von deren Routing Tables ersichtlich.

- Dass mehr Hubs gefunden werden, als physisch tatsächlich vorhanden sind, ist größtenteils auf virtuelle Maschinen zurückzuführen. Ein weiterer Grund dafür kann sein, wie auch zuvor beschrieben, wenn nach den Informationen der Switches bei einer Link Aggregation Verbindung auf der einen Seite die physischen Ports und der anderen Seite der virtuelle Port der Link Aggregation Group angegeben werden, wodurch ein Port auf der einen Seite mit mehreren Ports auf der anderen Seite verbunden ist (und dadurch per Definition ein Hub dazwischen liegen muss).

- Bei einigen Geräten wurde eine falsche Subnetzmaske gefunden, welche dazu führt, dass zu große, andere einschließende Subnetze erkannt wurden. Durch die vom Programm verwendete Methode, jede IP-Adresse jeweils dem kleinsten passenden Subnetz zuzuordnen, sind die Ergebnisse dennoch korrekt.

- Wie im vorigen Abschnitt kurz erwähnt, antworten einige Geräte auf jede beliebige SNMP Community, wobei manche in der Antwort die hingesendete Community und manch andere die Default Community `public` zurückschicken. Es gibt also Geräte, die mit einer anderen Community antworten, als mit der sie gefragt wurden.

- Weiters existieren Geräte, bei denen die Antwort von einer anderen IP-Adresse kommt als jene, an die die Anfrage gesendet wurde. Um diese zuordnen zu können, wird die IP-Adresse des abgefragten Gerätes in der Request ID kodiert mitgeschickt. Dieser und der vorige Punkt sind daher Gründe, warum es nicht nur ausreichend ist, irgendeine Antwort von einem Gerät zu erhalten, um es als SNMP-Gerät zu erkennen, sondern der Inhalt dieser Antwort muss auch ausgelesen und interpretiert werden.

- Manche Geräte senden von sich aus SNMP Traps. Um diese von Antworten auf Anfragen der Discovery-Maschine unterscheiden zu können, müssen diese Pakete ebenfalls analysiert werden, um sie verwerfen zu können.

- In manchen Fällen antworten Geräte auf eine Anfrage mit einer IP-Adresse, für welche sie aber, laut den über SNMP ausgelesenen Interface Tables, gar kein zugehöriges Interface besitzen. Die Ursache dafür sind fehlerhafte SNMP Engines, die inkorrekte Daten liefern. Der Algorithmus berücksichtigt dies jedoch und legt bei diesen Geräten entsprechende Interfaces an.

- Generell sind die Informationen aus den Interface Tables leider sehr unzuverlässig, was auf die Eigenarten in den SNMP-Implementierungen verschiedener Hersteller zurückzuführen ist. Oft enthalten diese Tabellen virtuelle Interfaces, die so nicht physisch existieren, manchmal ist es aber auch umgekehrt und (physische oder virtuelle) Interfaces, die ein Gerät offensichtlich besitzt, scheinen nicht auf.

- Es fällt auf, dass die MIBs zwar festlegen, welches Format die Variablen und Tabellen haben müssen, die ein Gerät über SNMP anbietet, jedoch ist nicht strikt definiert, mit welchen Daten diese Variablen und Tabellen gefüllt werden müssen. Dadurch kommt es, ähnlich wie im vorigen Punkt beschrieben, zu einem äußerst unterschiedlichen Verhalten einzelner Geräte im Bezug darauf, welche Informationen angeboten werden (auch wenn diese überall im selben Format sind). Dies muss entsprechend berücksichtigt werden.

- Sehr oft (meist bei Servern und Switches, dabei wiederum bei virtuellen Interfaces) kommt vor, dass laut den Interface Tables MAC-Adressen mehrfach an die Interfaces eines Geräts vergeben sind. Dies kann einzelne Interfaces (so haben bei manchen Switches z.B. ein physisches Interface und das virtuelle Management Interface die gleiche MAC-Adresse) oder auch alle Interfaces betreffen (z.B. verwendet der Legacy-Switch `SSRV3b` nur eine einzige MAC-Adresse). Manchmal kommt noch dazu, dass diese an mehrere Interfaces vergebenden MAC-Adressen an mehreren Ports gefunden werden (wenn z.B. ein Server mehrfach an einen oder mehrere Switches angeschlossen ist). Damit es bei der Bestimmung der Verbindungen der Geräte untereinander mittels der Informationen aus den AFTs zu keinem totalen Chaos kommt, verwendet der Algorithmus entsprechende Heuristiken (siehe Abschnitt *4.7.3 Topology Inference*).

- Beim Legacy-Switch `SSRV3b` ist weiters das in [76] beschriebene Problem aufgetreten, dass die Bytes in der Port ID beim STP von verschiedenen Switches bei der Ausgabe der Information über SNMP anders interpretiert werden. Dadurch konnte zwar `SSRV3b` auf die Ports anderer Switches verweisen, andere aber nicht auf dessen Ports. Da dieser im Spanning Tree jedoch ein Blatt ist, hat dies keine größeren Konsequenzen, außer dass ein über STP deaktivierter Link zu diesem Switch nicht erkannt wurde.

- Ein weiteres bei diesem Switch aufgetretenes Problem ist die Tatsache, dass die Einträge in dessen AFT ein Timeout von 60 Minuten haben (5 Minuten ist normalerweise der Standard). Das heißt, es dauert eine Stunde, bis der Switch merkt, dass ein Gerät nicht mehr im Netzwerk

ist. Daher wurden viele Geräte (z.B. Laptops), die offensichtlich abgeschaltet oder vom Netzwerk getrennt wurden, dennoch am Uplink Port dieses Switches gefunden, aber nirgends sonst (da die entsprechenden Einträge bei allen anderen Switches schon gelöscht worden waren). Theoretisch würde das bedeuten, dass diese Geräte an diesem Uplink Port angeschlossen wären, was aber offensichtlich nicht der Fall ist. Daher kommt auch die Entscheidung, im Algorithmus Geräte, die an Uplink Ports gefunden werden, zu ignorieren. Dies ist mit ein Grund, warum die in manch anderen Arbeiten (z.B. [76], [10], [45] oder [5]) vorgestellten Algorithmen – auch wenn sie unter ihren definierten, strikten Voraussetzungen korrekt sind – in der Praxis scheitern.

- Bei einer ganz speziellen Konstellation von Switches konnte die Einordnung in den Spanning Tree nicht erkannt werden. Bei den Enterasys-Geräten handelt es sich im Normalfall entweder um Verteiler, bei denen einzelne Switches an einer gemeinsamen Backplane angeschlossen sind. Dabei fungiert ein normaler Port eines Slots als Uplink Port (zu einem anderen Verteiler) und die anderen Slots (des gleichen Verteilers) verweisen über ihre Backplane Ports auf den Slot mit dem eigentlichen Uplink Port. Oder es handelt sich um einen Matrix Switch, welcher nur als ein einziges Gerät ansprechbar ist. In einem besonderen Fall gibt es jedoch einen Matrix Switch *und* einen dazugehörigen Slot, der separat ansprechbar ist. Da diese aber zusammengehören und der Matrix Switch die Verwaltung des Spanning Trees übernimmt, verfügt der einzelne Slot über keine STP-Informationen (da er nur Teil des größeren ist). Im Endeffekt handelt es sich aber um ein separates Gerät (das auch keine Hinweise darauf gibt, dass es Teil eines anderen Switches ist), das wegen der fehlenden STP-Daten nicht in den Spanning Tree eingeordnet werden kann. Über Umwege (den Inhalt der AFTs in denen auch das Management Interface des Slots enthalten ist) konnte das Programm ihn aber dennoch zumindest zum richtigen Port des Core-Switches, an dem der Matrix Switch hängt, zuordnen. Dies ist jedoch nicht ganz korrekt, da nun der Slot nicht als Teil des größeren Switches erkannt wurde, sondern beide als mit dem selben Uplink Port des Core-Switches verbunden dargestellt werden (wodurch auch ein virtueller Hub eingeführt wurde).

- Backplane Ports bedürfen grundsätzlich einer Spezialbehandlung, da nur einer von ihnen über STP als Uplink erkannt wird, wie im vorigen Punkt beschrieben wurde. Ein Slot hat aber mehrere Backplane Ports, nämlich einen zu jedem anderen Slot im Verteiler, die alle als Uplink fungieren. Auf ihnen werden daher alle Geräte, welche an den anderen Slots angeschlossen sind, gehört. Damit die Verbindungen zwi-

schen den Geräten korrekt erkannt werden können, müssen die AFT-Einträge der Backplane Ports ignoriert werden.

- Die speziellen Laborgeräte befinden sich im Netzwerk in einem eigenen VLAN, welches durch eine Firewall geschützt wurde und die jeglichen Verkehr von außen in dieses VLAN blockierte. Dennoch war es dem ARP Discovery über das Senden von VLAN getaggten Frames in früheren Tests möglich, diese Firewall zu umgehen. Dadurch wurde bewiesen, dass die entsprechende Technik funktioniert. Da diese Firewall inzwischen abgebaut wurde, konnte sie in den hier beschriebenen Szenarien nicht mehr berücksichtigt werden.

- Nur um auszuprobieren wie das Programm skaliert, wurde einmal das gesamte weltweite Novartis-Netzwerk gescannt (indem das Subnet Size Limit angehoben wurde) – auch wenn der Algorithmus und das Programm nur für den Scan eines LANs konzipiert wurden. Theoretisch hätten auch mit einem höheren Subnet Size Limit keine Subnetze, die nicht zum lokalen LAN gehören, gefunden werden dürfen. Durch falsch konfigurierten Subnetzmasken kam es aber zu Überlappungen mit den zu anderen LANs gehörigen Adressbereichen. Alternativ hätten die Subnetze aus anderen LANs auch einfach manuell eingetragen werden können. Der Scan des gesamten Netzwerks hat prinzipiell funktioniert, jedoch mussten dafür einige Schritte deaktiviert werden (vgl. *B.4 Bedienung*), weil sonst dem Programm der Speicher ausgegangen wäre. Das Programm ist also nicht ohne gewisse Anpassungen beliebig skalierbar. Es wurden insgesamt 950.730 IP-Adressen in 2.581 Subnetzen gescannt. Das Discovery benötigte etwa 36 Stunden. Diese lange Dauer liegt jedoch nicht an den Scans, sondern an der langsamen SNMP-Kommunikation über WAN-Links zu verschiedenen Kontinenten. Dabei wurden Timeouts vollständig ausgereizt und während das Auslesen einer Tabelle mit vielen tausend Einträgen über das LAN einige Sekunden dauert, so benötigte in diesem Fall das Lesen jeder einzelnen Zeile (!) mehrere Sekunden. Daher stammt die extrem lange Gesamtdauer. Weiters brauchten auch einige interne Berechnungen bzw. Hilfsoperationen (z.B. das Enumerieren von IP-Adressen) entsprechend mehr Zeit. Dies sind jedoch Aspekte der konkreten Umsetzung des Algorithmus' im Programm, die noch weiter optimiert werden könnten, um Laufzeit und Speicherverbrauch zu verbessern, jedoch keine Probleme des Algorithmus' an sich (genausowenig wie der langsame SNMP-Datenverkehr auf den Algorithmus zurückzuführen ist).

- Es wurde auch versucht, das Programm auf einer virtuellen Maschine (VMware) laufen zu lassen. Dies funktionierte prinzipiell, bis auf die Tatsache, dass VLAN getaggte Frames nicht korrekt versandt und empfangen werden konnten (trotz laut Anleitung korrekter Konfigura-

tion der VMware), wodurch das ARP Discovery von seiner Effektivität verlor.

Viele der zuvor genannten Punkte zeigen, dass eine Vielzahl von Details zu beachten sind. Durch die vielen Sonder- und Fehlerfälle sowie das unterschiedliche Verhalten einzelner Geräte ist es umso wichtiger, einen möglichst robusten Algorithmus zu verwenden – selbst wenn dies den Einsatz von Heuristiken bedeutet und zu Lasten der hundertprozentigen Korrektheit oder Vollständigkeit geht.

Diese Auflistung sollte vor allem dafür ein Bewusstsein schaffen, mit welcher Art von Problemen bei der Entwicklung eines Discovery-Systems für den realen Einsatz gerechnet werden muss.

## 6.5 Bewertung

Zusammenfassend kann gesagt werden, dass das Programm alle Tests bestanden hat. Die Ergebnisse können (nach Abgleich mit der Netzwerkdokumentation und dem NCL-System) als vollständig und korrekt angesehen werden, auch wenn sie nicht als hundertprozentig perfekt bezeichnet werden können (z.B. Erkennung von Hubs). Eine derartige Perfektion liegt aber – bei vollem Bewusstsein darüber – nicht im Rahmen des Möglichen des Algorithmus' und der von ihm verwendeten Daten. Eindeutig sind die Ergebnisse jedoch als voll tauglich für den praktischen Einsatz zu bezeichnen. Außerdem variieren die Ergebnisse nur im Bezug auf die Endgeräte, Infrastruktur und Topologie werden konsistent richtig erkannt.

Weiters hat das Programm seine Robustheit und Fähigkeit bewiesen, sowohl mit einem extrem heterogenen Umfeld (verschiedenste Geräteklassen von verschiedensten Herstellern in verschiedensten Modellen), als auch mit diversen typischerweise in einem modernen LAN eingesetzten Technologien (z.B. VLANs, VRRP/HSRP, Link Aggregation, ARP Proxies etc.) sowie mit Sonderfällen, aber auch mit Inkonsistenzen in den Daten und mit Konfigurationsfehlern umgehen zu können.

# Kapitel 7

# Einsatzszenarien

Im vorigen Kapitel wurde gezeigt, dass das entwickelte Programm für den praktischen Einsatz tauglich ist. Nun werden einerseits einige konkrete Anwendungsbespiele der vorliegenden Version, andererseits aber auch mögliche zukünftige Erweiterungen und deren Einsatzszenarien betrachtet.

## 7.1 Derzeitige Anwendungen

Obwohl das Programm eigentlich nur einen Proof-of-Concept darstellt, so kann es dennoch bereits auch produktiv angewendet werden. Folgende Liste – die keinerlei Anspruch auf Vollständigkeit erhebt – soll einige Anregungen für die Verwendung des Programms im Network Management bieten:

- Eine zentrale Funktion des Programms ist die Bestimmung der Topologie und der Netzwerkinfrastruktur. Dies kann dazu dienen, um schnell einen Überblick über ein Netzwerk zu erhalten – und zwar sowohl auf Layer 3, als auch auf Layer 2.
- Wird die bestimmte Topologie und gefundene Netzwerkinfrastruktur weiter im Detail betrachtet, so kann dies bei der Erstellung einer Netzwerkdokumentation helfen. Das Programm liefert dazu Informationen, wie z.B. welche Subnetze und VLANs es gibt, welche Router vorhanden sind und welcher davon wofür zuständig ist, wie der Spanning Tree aussieht, wie Switches untereinander verbunden sind usw.
- Die zweite Kernfunktionalität des Programms ist das Finden von Geräten im Netzwerk. Dies kann analog dazu dienen, um rasch an eine Übersicht über die im Netzwerk vorhandenen Geräte zu gelangen.
- Auch die Methoden, mit welcher einzelne Geräte gefunden wurden (oder nicht), kann interessante Hinweise geben (wenn z.B. Geräte mit dem *ICMP Discovery*, nicht aber mit dem *ARP Discovery* gefunden

wurden oder umgekehrt). Ein bei den Tests begegneter konkreter Fall ist etwa, dass sehr viele Geräte nur mit dem *DNS Discovery* gefunden wurden, deren Existenz aber sonst nicht bestätigt werden konnte. Dies könnte zum Beispiel auf veraltete Einträge im DNS hinweisen.

- Weiters können die mit dem Programm bestimmten Daten eine Grundlage für eine Inventarisierung darstellen.
- Im Bezug auf die Inventarisierung kann das Programm besonders dabei helfen, meist statische Asset-Daten (z.B. Seriennummern) mit dynamischen Netzwerkdaten (z.B. IP-Adressen) abzugleichen.
- Auch beim Management der Netzwerksicherheit kann das Programm behilflich sein. So liefert es beispielsweise eine Übersicht über alle aktiven SNMP-Geräte, inklusive der SNMP Communities unter denen sie ansprechbar sind. Dies kann bei der Identifikation von Sicherheitslücken seinen Beitrag leisten. So kann etwa in einem Netzwerk die Richtlinie gelten, dass keine SNMP-Geräte mit der Default Community `public` ansprechbar sein dürfen. Das Programm hilft dabei, dies auch zu überprüfen und Geräte zu finden, die nicht dieser Richtlinie entsprechen.
- Analog dazu kann das Programm zur Suche nach „unerwünschten“ Geräten verwendet werden, d.h. Geräte, die von Nutzern unerlaubterweise und eigenmächtig an das Netzwerk angeschlossen wurden.
- Eine weitere wichtige Funktion des Programms ist die Suche nach einem oder mehreren bestimmten Geräten (nach verschiedensten Kriterien) und der Ausgabe der zu diesen Geräten gefundenen Informationen (z.B. IP-Adressen, Netzwerknamen, VLANs etc.).
- Eine besonders wichtige Information zu einem Gerät ist dessen Standort. Das Programm kann zwar genau genommen nur den Anschlussort (Port $a$ von Switch $b$) eines Geräts feststellen (was schon einmal besser als nichts ist), wenn dazu aber noch anderweitig Patch-Daten (Port $a$ von Switch $b$ ist zur Dose $c$ in Raum $d$ von Gebäude $e$ gepatcht) vorhanden sind, so können diese korreliert und somit auch der tatsächliche physische Standort eines Geräts bestimmt werden. Dies kann beispielsweise bei der Suche nach „verschollenen“ Geräten behilflich sein.
- Die vom Programm gesammelten Informationen betreffen jedoch nicht nur Endgeräte, sondern auch Infrastrukturkomponenten. So kann etwa eine Übersicht über einen bestimmten Switch – welche Ports dieser hat, deren Status (up/down, Geschwindigkeit), an ihnen angeschlossene Geräte, deren VLAN-Konfiguration etc. – angezeigt werden.

- Analoges gilt auch für den Layer 3. So kann etwa eine Routing-Übersicht dargestellt werden, die zeigt, welcher Router Routen zu welchen Subnetzen besitzt.
- Weiters können all die vom Programm gesammelten Informationen auch zum Erkennen von Konfigurationsfehlern ausgenutzt werden. Einige bei den Test konkret begegnete Fälle sind z.B. auf verschiedenen Switches inkonsistent definierte VLAN-Namen, Fehler in der VLAN-Konfiguration von Switch Ports, falsche Subnetzmasken oder fehlende bzw. inkorrekte SNMP-Informationen.
- Die bestimmten Informationen lassen sich jedoch nicht nur abfragen oder auflisten, sondern es können damit auch statistische Auswertungen (z.B. Anzahl an Geräten eines bestimmten Typs, Anzahl an Geräten in einem bestimmten VLAN oder Subnetz, Anzahl an SNMP-Geräten, die mit einer bestimmten Community ansprechbar sind, Anzahl an freien, belegten up oder down Ports eines bestimmten Switches etc.) gemacht werden, die ebenfalls dabei helfen, einerseits ein besseres Bild über das Netzwerk zu erhalten und andererseits Fehler zu erkennen.
- Seine volle Stärke könnte das Programm dann ausspielen, wenn es mit anderen Daten bzw. Systemen (z.B. Netzwerk Monitoring, Asset Management, Configuration Management der Switches und Router etc.) verknüpft bzw. integriert werden würde. Dann könnte beispielsweise der zuvor angesprochene Datenabgleich zur Standortbestimmung eines Geräts automatisch geschehen. Die dafür nötigen Datenanbindungen müssten aber erst erstellt werden (siehe dazu auch Abschnitt *7.2 Erweiterungen*).

Diese Punkte sind nur einige Ideen aus einer Fülle von Möglichkeiten, die alle den Alltag eines Netzwerk-Administrators erleichtern können. Die besondere Stärke des Programms liegt darin, dass verschiedene Daten einzeln erfasst werden, danach aber beliebig miteinander verknüpft werden können, sodass der Fantasie keine Grenzen gesetzt sind, wenn es darum geht, Abfragen zu entwickeln, bestimmte Informationen zu extrahieren oder spezifische Aspekte des Netzwerks zu beleuchten.

## 7.2 Erweiterungen

Auch wenn das Programm, wie gezeigt wurde, durchaus in seiner jetzigen Form praktisch eingesetzt werden kann, so ist es doch nicht perfekt. Es gibt einige im Programm nicht berücksichtige Punkte bzw. Techniken sowie eine Vielzahl von denkbaren Erweiterungen, von denen einige hier angesprochen werden sollen.

Da das Network Discovery nur einen Teil des Network Managements ausmacht, aber als Basis für andere Teile davon dienen kann, liegt, wie im vorigen Kapitel angesprochen, ein Zusammenschluss von einzelnen Systemen, die jeweils eine Teilaufgabe aus dem Network Management übernehmen, nahe. Folgende Liste betrachtet einige Punkte dazu:

- Das Programm könnte als eine Komponente in einer kompletten *Network Management Suite* – bestehend beispielsweise aus *Network Discovery* (Finden von Geräten und Bestimmen der Topologie), *Network Monitoring* (Überwachen der gefundenen Geräte), *Asset Management* (Inventarisierung der Geräte) und *Configuration Management* (z.B. Verwaltung der Patch-Daten von Switches) – eingesetzt werden.

- Der Vorteil in der Anbindung des Network Discoverys an das Asset Management liegt darin, dass statisch erfasste Inventardaten (z.B. Seriennummern, Leasingvertrags-Daten etc.) mit dynamischen, aus dem Netzwerk ausgelesenen Daten (z.B. IP-Adressen, MAC-Adressen, Netzwerknamen etc.) kombiniert werden und so alle zu einem Gerät verfügbaren Daten gesammelt an einer Stelle abgefragt werden könnten.

- Das im vorigen Kapitel gebrachte Beispiel der Standortbestimmung zeigt, wie durch die Verknüpfung einzelner Daten – den Patchungen von Switch Ports (aus dem Configuration Management) und dem Anschlussort von Geräten (aus dem Network Discovery) – neue Informationen (nämlich der physische Standort von Geräten) mit einem echten Mehrwert entstehen können. Zusätzlich könnten auch weitere, nicht direkt mit dem Network Management zusammenhängende Daten, wie etwa Gebäude- oder Mitarbeiterdaten, miteinbezogen werden. So könnte beispielsweise jedem Gerät auch ein Besitzer bzw. Verantwortlicher zugewiesen werden, für den Daten – wie z.B. seine Telefonnummer – aus den angesprochenen anderen Quellen bezogen werden.

- Weiters kann das Network Discovery Vorgaben für das Network Monitoring liefern (etwa Listen von zu überwachenden Geräten), welches wiederum beispielsweise zur Erkennung von Fehlerzuständen (z.B. Ausfälle von Geräten oder Verbindungen) oder Konfigurationsfehlern (was auch im Zusammenhang mit dem Configuration Management dienen kann, welches die korrekten Einstellungen vorgibt) steht. Auch können Inkonsistenzen in Daten automatisch erkannt werden. Ist beispielsweise ein Switch Port als nicht gepatcht markiert, wird aber daran angeschlossen ein Gerät gefunden, so ist dies ein Hinweis auf einen Fehler in den Patch-Daten (welche manuell geführt werden müssen und daher fehleranfällig sind).

Der generelle Gedanke hinter den genannten Punkten ist es, ein einziges System zu haben, das alle Aspekte des Network Managements vereint. Dies vereinfacht nicht nur die Arbeit, da alle Informationen an einer Stelle abrufbar sind, sondern auch allein aus der Verknüpfung von Daten können bereits, wie gezeigt, neue Informationen und ein Mehrwert entstehen.

Unabhängig vom Einsatz des Programms im Zusammenhang mit anderen, kann es auch für sich betrachtet verbessert und erweitert werden. Ein erster Ansatzpunkt dazu wäre das Discovery an sich zu verfeinern und auszubauen. In folgender Liste werden einige Ideen dafür behandelt:

- Derzeit kann das Discovery immer jeweils nur über ein Interface der Discovery-Maschine ausgeführt werden. Besitzt die Maschine jedoch mehrere Interfaces, so würde ein Discovery über verschiedene Interfaces es dem Programm auch erlauben, das Netzwerk von unterschiedlichen Perspektiven aus zu betrachten. Dies wäre besonders hilfreich, wenn es sich aus der Struktur des Netzwerkes (z.B. auf Grund von Firewalls) ergibt, dass der Ort, von dem aus das Discovery stattfindet, eine Relevanz im Bezug auf die Ergebnisse besitzt. Das Discovery über mehrere Interfaces auszuführen, wäre einfach und würde von dem Programm in seiner jetzigen Form bereits unterstützt werden. Schwieriger jedoch wäre es, die gefundenen – und möglicherweise abweichenden – Daten danach in einem Modell zu konsolidieren.

- Die zuvor genannte Idee lässt sich insofern erweitern, dass das Discovery nicht nur über mehrere Interfaces einer Maschine durchgeführt werden könnte, sondern überhaupt von verschiedenen Maschinen aus, wodurch ein echtes verteiltes Discovery stattfinden würde. Hierbei könnten nicht nur insgesamt mehr Daten erfasst werden (falls gewisse Bereiche des Netzwerkes nur von bestimmten Maschinen aus einsehbar sind), sondern auch aus der Diskrepanz der Daten der einzelnen Maschinen ließen sich Rückschlüsse auf das Netzwerk ziehen. Bei einem derartigen Vorgehen müssten die Daten der verschiedenen Maschinen an einer zentralen Stelle gesammelt und danach ebenfalls konsolidiert werden. Die Architektur des Programms trennt derzeit bereits logisch zwischen den Discovery-Funktionen und allen anderen, sodass es einfach wäre, das Discovery auch physisch auf mehrere Maschinen aufzuteilen. Das Zusammenfassen der Daten müsste jedoch noch implementiert werden.

- Der genannte Mechanismus zum verteilten Discovery ließe sich auch zur Überwachung von mehreren LANs einsetzen, indem eine Discovery-Maschine pro LAN eingesetzt wird und die Daten aus allen LANs an einer Stelle gesammelt werden. Hierbei muss aber im Datenmodell beachtet werden, dass gewisse Dinge nicht konsistent sein

müssen (etwa die Definition von VLANs), da es sich um separate LANs handelt.

- Momentan werden nur Snapshots des Netzwerkes erfasst und bereits zuvor gesammelte Daten werden bei einem neuen Discovery verworfen. Durch das Hinzufügen einer Zeit-Komponente im Datenmodell, könnten auch History-Funktionen angeboten und die Veränderung des Netzwerkes über die Zeit (z.B. ob und wie Geräte bewegt wurden) betrachtet werden. Für diese Änderung wären keine Adaptionen des Discovery-Algorithmus' an sich vonnöten, jedoch des Datenmodells und der Auswertungsfunktionen.

- In der jetzigen Form verwendet das Programm nur das Default Gateway als Router über den Daten in andere Subnetze geschickt werden. Es könnten jedoch auch andere Router berücksichtigt werden, falls auf der Discovery-Maschine explizit Routen definiert sind.

Während oben genannte Punkte konzeptuelle Verbesserungen des momentanen Discovery-Ansatzes betreffen, könnten auch konkrete Methoden verbessert oder komplett neue Techniken und Ansätze implementiert werden (durch die modulare Architektur des Programms wäre es einfach, neue Discovery-Module hinzuzufügen):

- Ein Ansatz wäre beispielsweise, mehr Techniken zum Finden von Geräten zu verwenden. Dies kann aktive (z.B. *TCP SYN Scan*, *TCP ACK Scan*, *TCP Echo*, *UDP Ping*, *UDP Echo*, *DNS Zone Transfer*, *UPnP*, *LLDP*, *CDP* etc.) und passive Methoden (Abhören von generellem Traffic, *SMB Announcements*, *Routing-Protokollen*, *Discovery-Protokollen* etc.) beinhalten.

- Ein anderer Ansatz wäre, mehr Informationen zu den gefundenen Geräten zu bestimmen und etwa *WMI*, *Remote Shells*, *OS Fingerprinting*, *Konfigurations-Dateien* etc. miteinzubeziehen.

- Eine besonders interessante Information über Geräte sind die Services, die sie anbieten. Diese könnten einerseits etwa über SNMP ausgelesen oder auch mit einigen der zuvor genannten Methoden bestimmt, aber andererseits auch mittels *TCP Port Scans* und *UDP Port Scans* festgestellt werden.

- Eine weitere, derzeit überhaupt nicht berücksichtigte Information betrifft die Benutzer von Geräten. Diese könnten z.B. über *Systemaufrufe* bestimmt werden (d.h. welcher User gerade auf einer bestimmten Maschine eingeloggt ist). Beim Umgang mit derartigen Informationen sollten jedoch – neben dem rein Technischen – auch Datenschutzaspekte bedacht werden.

- Momentan wird für jeden Switch Port nur das Default VLAN (das ist dasjenige, mit dem eingehende, ungetaggte Frames getagged werden) festgestellt. Die VLAN-Konfiguration könnte aber noch detaillierter bestimmt werden, indem etwa auch Ingress-, Egress- und Filtering-Listen ausgelesen werden.

- Derzeit wird zwar Link Aggregation insofern unterstützt, als dass die je einer Link Aggregation Group zugeordneten virtuellen Ports wie normale Ports behandelt werden, es könnten jedoch auch noch mehr Informationen bestimmt werden, wie beispielsweise welche physischen Ports zu welcher Link Aggregation Group gehören.

- Weiters könnte versucht werden (etwa über SNMP) auch Hinweise über die Übertragungsmedien (z.B. ob es sich in einem bestimmten Fall um ein Glasfaser- oder Kupferkabel handelt) zu erhalten.

- In seiner jetzigen Form erkennt das Programm zwar virtuelle Router über deren MAC-Adressen, es erfolgt jedoch keine Zuordnung zwischen diesen und den physischen Routern. Dafür müssten entsprechende Informationen über SNMP ausgelesen werden.

- Analog dazu werden auch virtuelle Maschinen (vom Typ *VMware*) über ihre MAC-Adresse erkannt, es wird jedoch keine Zuordnung von VM Hosts und Guests getroffen. Hierzu wären spezielle Heuristiken zu entwickeln und zu implementieren.

- Da der vorgestellte Algorithmus stark auf dem Einsatz von SNMP zum Erheben von vielen Daten basiert, würde es nahe liegen, diese Abhängigkeit von SNMP zu reduzieren, indem etwa gewisse Daten auch anders erfasst werden. So könnte beispielsweise versucht werden, Routing-Informationen mittels *Traceroute* zu bestimmen usw.

- Gerade beim zuvor genannten Traceroute, aber auch bei vielen anderen Techniken, würde sich der Einsatz von *Source Routing* anbieten, da dies erlauben würde, trotz des immer selben physischen Ortes, von dem aus das Discovery durchgeführt wird, Pakete so umzuleiten, dass verschiedene Perspektiven im Discovery möglich werden (sofern Source Routing nicht aus Sicherheitsgründen deaktiviert ist).

- Derzeit geht der Algorithmus von einem statischen Netzwerk aus, dessen Zustand sich während des Discoverys nicht verändert. Ein Ansatz wäre, auch eine dynamische Umgebung zu berücksichtigen. Die Annahme eines während des Discoverys unveränderlichen Netzwerkes wird jedoch auch meist in der Literatur zu diesem Thema (z.B. in [76] und [10]) gemacht. Daher wäre in diesem Bereich erst einmal Forschung zu betreiben und die Entwicklung entsprechender Modelle und Algorithmen nötig.

Da das Programm in seiner jetzigen Form, wie bereits mehrfach erwähnt, nur ein Proof-of-Concept ist, wurden bewusst einige Funktionen – vor allem betreffend Usability und dergleichen – weggelassen, um es nicht unnötig zu verkomplizieren oder vom Wesentlichen abzulenken. Aber für den produktiven Einsatz wären Erweiterungen in diesem Bereich empfehlenswert. Folgende Liste bietet dazu einige Ansatzpunkte:

- Derzeit wird vom Programm nur ein Single-User-Betrieb unterstützt. Durch die modulare Architektur des Programms, könnten einzelne Komponenten jedoch auch physisch verteilt werden (und etwa über *RMI* kommunizieren, wodurch nur minimale Anpassungen vonnöten wären). So wäre es etwa denkbar $n$ Clients (mit je einem User Interface) zu haben, die alle mit einem zentralen Core-Server kommunizieren, der die Anfragen der Clients behandelt, die Datenbank verwaltet (die sich selbst wieder auf einer anderen Maschine befinden könnte) sowie den Discovery-Prozess steuert, welcher wiederum von $m$ verschiedenen Discovery-Servern aus durchgeführt wird. In einer derartigen physischen Architektur (die bereits von der bestehenden logischen Architektur unterstützt wird) wären sowohl ein verteiltes Discovery, als auch ein Multi-User-Betrieb abgebildet.

- Momentan wird, der Einfachheit halber, SNMP-Kommunikation nur über den dazugehörigen Default Port 161 abgewickelt, welcher fix eingestellt ist. Diesen änderbar zu machen (falls bestimmte Geräte einen anderen Port verwenden) wäre jedoch ein Leichtes, da alle Module und sonstige Komponenten bereits variable Port-Nummern unterstützen. Daher wäre einzig eine Erweiterung des User Interfaces nötig.

- Weiters wird derzeit nur Kommunikation über *SNMPv1* unterstützt, was zu Problemen führen könnte, wenn etwa Switches und Router (z.B. aus Sicherheitsgründen) nur über höhere SNMP-Versionen ansprechbar sind. Eine Unterstützung von *SNMPv2* und *SNMPv3* wäre jedoch leicht möglich, da alle SNMP-Kommunikation über die Bibliothek *SNMP4J* abgewickelt wird, welche diese SNMP-Versionen bereits anbietet. Daher wären an den Discovery-Modulen und anderen Komponenten keine signifikanten Änderungen nötig.

- Analog dazu verwendet das Programm derzeit nur *IPv4*. Eine Unterstützung von *IPv6* wäre grundsätzlich möglich, da zum direkten Zugriff auf das Netzwerk die Bibliothek *Jpcap* verwendet wird, welche ihre Funktionen prinzipiell auch für IPv6 anbietet. Es müssten jedoch sowohl die Bibliothek selber (im Bezug auf den Umgang mit VLANs), als auch alle Discovery-Module entsprechend angepasst werden, vor allem da IPv6-Adressen einem anderen Format folgen.

- Derzeit ist das Programm lediglich auf *Microsoft Windows* einsetzbar. Daher liegt eine Portierung auf andere Plattformen nahe. Dazu müsste einerseits die Bibliothek `Jpcap.dll` für die jeweils entsprechende Plattform kompiliert werden und andererseits einige Plattformspezifische Techniken, die das Programm verwendet (z.B. Auslesen des Default Gateways über das Ausführen des Systemkommandos `ipconfig`), überarbeitet werden. Alle anderen Komponenten sind entweder bereits Plattform-unabhängig (z.B. *SNMP4J*) oder auch für andere Plattformen verfügbar (z.B. *Graphviz*).

- Schließlich wäre auch eine Reihe von Verbesserungen des Clients denkbar, etwa Query-Funktionen, für die keine SQL-Kenntnisse notwendig sind, erweiterte Mapping-Funktionen und vieles mehr.

# Kapitel 8

# Zusammenfassung

Nach der Beschreibung der konkreten Problemstellung hinter dem Thema *Network Discovery* – das Auffinden von Geräten in einem Computernetzwerk und von Informationen sowohl über diese selbst, als auch über deren Verbindungen untereinander – und dessen praktischer Relevanz – die aus dem Network Discovery gewonnen Daten können in vielen Aufgaben des Network Managements verwendet werden – wurde zuerst eine Einführung in Computernetzwerke geliefert. Diese beschäftigte sich mit allgemeinen Grundlagen, dem OSI-Modell, Netzwerkkomponenten, Topologien, Ethernet, TCP/IP und einigen weiteren für das Discovery besonders relevanten Protokollen.

Ausgestattet mit diesem Basiswissen, wurden sowohl grundlegende Aspekte und Ansätze des Network Discoverys vorstellt (z.B. aktiv. vs. passiv, Probing vs. Polling und zentral vs. verteilt), als auch konkrete Discovery-Methoden beschrieben. Dabei wurden Techniken, die auf SNMP, ARP, ICMP, TCP, UDP und DNS basieren, im Detail vorgestellt. Einige weitere Methoden bzw. Datenquellen wurden ebenfalls präsentiert. Nach einer separaten Behandlung von wichtigen Sonderfällen, wurden die Techniken noch einmal zusammenfassend betrachtet, verglichen und somit ein Leitfaden zur deren Auswahl bei der Entwicklung eines Discovery-Algorithmus' geliefert.

Um darzustellen, wie diese Entwicklung konkret geschehen kann, wurde ein kompletter Algorithmus – mit Fokus auf ein Discovery bei minimalen Vorkenntnissen – vorgestellt, der einige der behandelten Techniken zusammenfasst und kombiniert. Dieser Algorithmus benötigt als Eingabedaten einzig die im Netzwerk verwendeten SNMP Communities und besteht aus drei wesentlichen Teilen. Als Erstes wird die Infrastruktur (Switches, Router, VLANs und Subnetze) bestimmt. Danach werden die gefundenen VLANs und Subnetze mit verschiedenen Methoden nach Geräten durchsucht. Abschließend wird die Topologie des Netzwerkes bestimmt, wofür von

den Switches zusätzliche Informationen abgerufen werden, aus denen dann die Topologie errechnet wird.

Der vorgestellte Algorithmus wurde weiters in dem eigens für diese Arbeit entwickelten Programm *Network Explorer* umgesetzt, welches ebenfalls ausführlich beschrieben wurde. Es handelt sich dabei um eine Java-Applikation, die über eine modulare Architektur verfügt und auf diverse Open-Source-Bibliotheken für bestimmte Aufgaben zurückgreift. Es wurde gezeigt, wie aus einzelnen Methoden in mehreren Schritten – zuerst als Algorithmus am Papier, dann in Form von Software – ein komplettes System erstellt werden kann.

Das entwickelte System wurde in einer realen Umgebung eingesetzt und konnte so seine Praxistauglichkeit unter Beweis stellen. Die Versuchsumgebung und der Versuchsaufbau wurden vorgestellt, ebenso wie die bei den Experimenten erzielten Ergebnisse, welche analysiert und bewertet wurde. Besondere Beobachtungen, die in diesem Zusammenhang gemacht wurden, wurden ebenfalls nicht vorenthalten.

Weiters wurden Szenarien präsentiert, wie die Software produktiv eingesetzt werden könnte und Vorschläge gemacht, wie das bestehende Programm noch erweitert und verbessert werden könnte. Im Laufe dieser Arbeit stand immer der praktische Nutzen der vorgestellten Themen im Vordergrund, wobei speziell Sonderfälle genannt und auf beachtenswerte Punkte hingewiesen wurde.

Abschließend hofft der Autor dieser Arbeit, dass er einen Einblick in dieses spannende Thema gewähren, ein Beispiel und eine Hilfestellung zur Umsetzung von Discovery-Systemen bieten sowie ein nützliches Werkzeug zur Verfügung stellen konnte.

# Anhang A

# Literaturverzeichnis


[1] Adhikari, A., L. Denby, J. Meloche und B. Rao: *Operational Layer-3 Topology*. Technischer Bericht, Avaya Labs Research, Avaya Inc., Basking Ridge, 2003. Auch verfügbar unter `http://citeseer.ist.psu.edu/678894.html` (20.07.2009).

[2] Arkin, O.: *Network Scanning Techniques*. Verfügbar unter `http://deathmule.nullfile.com/tools!/library/auditing/Network_Scanning_Techniques.pdf` (20.07.2009).

[3] Augustin, B., X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien und R. Teixeira: *Avoiding Traceroute Anomalies with Paris Traceroute*. In: *Proceedings of the ACM SIGCOMM*, Seiten 153–158, 2006. Auch verfügbar unter `http://doi.acm.org/10.1145/1177080.1177100` (20.07.2009).

[4] Beerliova, Z., F. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihal'ak und L. Ram: *Network Discovery and Verification*. IEEE Journal on Selected Areas in Communications, 24(12):2168–2181, 2006. Auch verfügbar unter `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4016133` (20.07.2009).

[5] Bejerano, Y., Y. Breitbart, M. Garofalakis und R. Rastogi: *Physical Topology Discovery for Large Multi-Subnet Networks*. In: *Proceedings of the IEEE INFOCOM*, Seiten 342–352, 2003. Auch verfügbar unter `http://www.ieee-infocom.org/2003/papers/09_02.pdf` (20.07.2009).

[6] Bellovin, S.: *A Technique for Counting NATted Hosts*. In: *Proceedings of the ACM SIGCOMM*, Seiten 267–272, 2002. Auch verfügbar unter `http://www.imconf.net/imw-2002/imw2002-papers/112.ps.gz` (20.07.2009).

[7] Bergman, R., H. Lewis und I. McDonald: *Printer MIB v2*. RFC 3805, Internet Engineering Task Force, 2004. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc3805.txt` (20.07.2009).

[8] Black, R., A. Donnelly und C. Fournet: *Ethernet Topology Discovery without Network Assistance*. In: *Proceedings of the IEEE International Conference on Network Protocols*, 2004. Auch verfügbar unter `http://research.microsoft.com/~austind/papers/icnp-topo-disc.pdf` (20.07.2009).

[9] Braden, R.: *Requirements for Internet Hosts - Communication Layers*. RFC 1122, Internet Engineering Task Force, 1989. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc1122.txt` (20.07.2009).

[10] Breitbart, Y., M. Garofalakis, C. Martin, R. Rastogi und A. Silberschatz: *Topology Discovery in Heterogeneous IP Networks: The NetInventory System*. In: *Proceedings of the IEEE INFOCOM*, Seiten 265–274, 2000. Auch verfügbar unter `http://citeseer.ist.psu.edu/breitbart00topology.html` (20.07.2009).

[11] Cai, T., P. Leach, Y. Gu und Y. Goland: *Simple Service Discovery Protocol*. Internet-Draft, Internet Engineering Task Force, 1999. Auch verfügbar unter `http://tools.ietf.org/id/draft-cai-ssdp-v1-00.txt` (20.07.2009).

[12] Case, J., M. Fedor, M. Schoffstall und C. Davin: *Simple Network Management Protocol (SNMP)*. RFC 1157, Internet Engineering Task Force, 1990. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc1157.txt` (20.07.2009).

[13] Cerf, V.: *ASCII Format for Network Interchange*. RFC 20, Internet Engineering Task Force, 1969. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc20.txt` (20.07.2009).

[14] Cisco Systems Inc.: *CDP Packet Format*. Verfügbar unter `http://www.cisco.com/univercd/cc/td/doc/product/lan/trsrb/frames.htm#xtocid12` (20.07.2009).

[15] Cisco Systems Inc.: *CISCO-HSRP-MIB*. Verfügbar unter `ftp://ftp.cisco.com/pub/mibs/v2/CISCO-HSRP-MIB.my` (20.07.2009).

[16] Cisco Systems Inc.: *Network Management Basics*. Verfügbar unter `http://www.cisco.com/en/US/docs/internetworking/technology/handbook/NM-Basics.pdf` (20.07.2009).

[17] Cisco Systems Inc.: *Switch Virtual Interface for Cisco Integrated Services Routers.* Verfügbar unter `http://www.cisco.com/en/US/prod/collateral/routers/ps5853/prod_white_paper0900aecd8064c9f4.pdf` (20.07.2009).

[18] Coates, M., R. Castro, R. Nowak, M. Gadhiok, R. King und Y. Tsang: *Maximum Likelihood Network Topology Identification from Edge-based Unicast Measurements.* SIGMETRICS Performance Evaluation Review, 30(1):11–20, 2002. Auch verfügbar unter `http://doi.acm.org/10.1145/511399.511337` (20.07.2009).

[19] Coates, M., M. Rabbat und R. Nowak: *Merging Logical Topologies Using End-to-end Measurements.* In: *Proceedings of the ACM SIGCOMM*, Seiten 192–203, 2003. Auch verfügbar unter `http://doi.acm.org/10.1145/948205.948230` (20.07.2009).

[20] Dall'Asta, L., I. Alvarez-Hamelin, A. Barrat, A. Vazquez und A. Vespignani: *Exploring Networks with Traceroute-like Probes: Theory and Simulations.* Theoretical Computer Science, 355(1):6–24, 2006. Auch verfügbar unter `http://alexves.googlepages.com/Theocompsci.pdf` (20.07.2009).

[21] Donnet, B., P. Raoult, T. Friedman und M. Crovella: *Efficient Algorithms for Large-Scale Topology Discovery.* In: *Proceedings of the ACM SIGMETRICS*, Seiten 327–338, 2005. Auch verfügbar unter `http://doi.acm.org/10.1145/1064212.1064256` (20.07.2009).

[22] Eriksson, B., P. Barford und R. Nowak: *Network Discovery from Passive Measurements.* In: *Proceedings of the ACM SIGCOMM*, Seiten 291–302, 2008. Auch verfügbar unter `http://doi.acm.org/10.1145/1402958.1402992` (20.07.2009).

[23] Govindan, R. und H. Tangmunarunkit: *Heuristics for Internet Map Discovery.* In: *Proceedings of the IEEE INFOCOM*, Seiten 1371–1380, 2000. Auch verfügbar unter `ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/99-717.ps.Z` (20.07.2009).

[24] Gunderson, S.: *Global IPv6 Statistics.* Verfügbar unter `http://www4.ietf.org/proceedings/08nov/slides/v6ops-4.pdf` (20.07.2009), 2004.

[25] Gunes, M. und K. Sarac: *Inferring Subnets in Router-level Topology Collection Studies.* In: *Proceedings of the ACM SIGCOMM*, Seiten 203–208, 2007. Auch verfügbar unter `http://doi.acm.org/10.1145/1298306.1298334` (20.07.2009).

[26] He, Y., G. Siganos, M. Faloutsos und S. Krishnamurthy: *Lord of the Links: A Framework for Discovering Missing Links in the Internet Topology*. IEEE/ACM Transactions on Networking, 17(2):391–404, 2009. Auch verfügbar unter `http://dx.doi.org/10.1109/TNET.2008.926512` (20.07.2009).

[27] Hinden, E.: *Virtual Router Redundancy Protocol (VRRP)*. RFC 3768, Internet Engineering Task Force, 2004. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc3768.txt` (20.07.2009).

[28] Huffaker, B., D. Plummer, D. Moore und K. Claffy: *Topology Discovery by Active Probing*. Technischer Bericht, Cooperative Association for Internet Data Analysis, University of California, San Diego, 2002. Auch verfügbar unter `http://www.caida.org/outreach/papers/2002/SkitterOverview/skitter_overview.ps.gz` (20.07.2009).

[29] IEEE 802.1ab Working Group: *Station and Media Access Control Connectivity Discovery*. Standard 802.1AB, Institute of Electrical and Electronics Engineers, 2005. Auch verfügbar unter `http://standards.ieee.org/getieee802/download/802.1AB-2005.pdf` (20.07.2009).

[30] IEEE 802.1d Working Group: *Media Access Control (MAC) Bridges*. Standard 802.1D, Institute of Electrical and Electronics Engineers, 2004. Auch verfügbar unter `http://standards.ieee.org/getieee802/download/802.1D-2004.pdf` (20.07.2009).

[31] IEEE 802.1q Working Group: *Virtual Bridged Local Area Networks*. Standard 802.1Q, Institute of Electrical and Electronics Engineers, 2003. Auch verfügbar unter `http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf` (20.07.2009).

[32] IEEE 802.3 Working Group: *LAN/MAN CSMA/CD Access Method*. Standard 802.3, Institute of Electrical and Electronics Engineers, 2008. Auch verfügbar unter `http://standards.ieee.org/getieee802/802.3.html` (20.07.2009).

[33] IEEE 802.3ad Working Group: *IEEE 802.3ad MIB*. Verfügbar unter `http://www.ieee802.org/3/publication/ad/IEEE8023-LAG-MIB.txt` (20.07.2009).

[34] International Organization for Standardization: *Open Systems Interconnection – Basic Reference Model: The Basic Model*. International Standard ISO/IEC 7498-1:1994, 1994. Auch verfügbar unter `http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip` (20.07.2009).

[35] International Telecommunication Union: *Abstract Syntax Notation One (ASN.1): Specification of basic notation*. Recommendation X.680, 2008. Auch verfügbar unter `http://www.itu.int/rec/T-REC-X.680/en` (20.07.2009).

[36] Jewell, B. und D. Chuang: *Definitions of Managed Objects for the Virtual Router Redundancy Protocol*. RFC 2787, Internet Engineering Task Force, 2000. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc2787.txt` (20.07.2009).

[37] Karacali, B. und M. Karol: *Network-wide Inference of End-to-End Path Intersections*. In: *Proceedings of the IEEE Network Operations and Management Symposium*, Seiten 168–175, 2008. Auch verfügbar unter `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4575131` (20.07.2009).

[38] Kessler, G. und S. Shepard: *A Primer On Internet and TCP/IP Tools and Utilities*. RFC 2151, Internet Engineering Task Force, 1997. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc2151.txt` (20.07.2009).

[39] Kreutzkamp, J., L. Hagge, E. Deffur, A. Gellrich und B. Schulz: *Experience with an IT Asset Management System*. In: *Proceedings of the Conference for Computing in High Energy and Nuclear Physics*, 2003. Auch verfügbar unter `http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUDT001.PDF` (20.07.2009).

[40] Lazar, S., D. Sidhu, S. Kodeswaran und R. Varadharaj: *ATM Network Discovery Using Mobile Agents*. In: *Proceedings of the Conference on Local Computer Networks*, Seiten 98–105, 1999. Auch verfügbar unter `http://ieeexplore.ieee.org/stamp/stamp.jsp?isnumber=17401&arnumber=802003` (20.07.2009).

[41] Levi, D. und D. Harrington: *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering, and Virtual LAN Extensions*. RFC 4363, Internet Engineering Task Force, 2006. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc4363.txt` (20.07.2009).

[42] Li, T., B. Cole, P. Morton und D. Li: *Cisco Hot Standby Router Protocol (HSRP)*. RFC 2281, Internet Engineering Task Force, 1998. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc2281.txt` (20.07.2009).

[43] Low, C.: *CISCO-VLAN-MEMBERSHIP-MIB*, 1998. Verfügbar unter `ftp://ftp.cisco.com/pub/mibs/v2/CISCO-VLAN-MEMBERSHIP-MIB.my` (20.07.2009).

[44] Lowekamp, B., D. O'Hallaron und T. Gross: *Direct Queries for Discovering Network Resource Properties in a Distributed Environment*. Cluster Computing, 3(4):281–291, 2000. Auch verfügbar unter `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/cmcl/archive/Remulac-papers/hpdc99-lowekamp.pdf` (20.07.2009).

[45] Lowekamp, B., D. O'Hallaron und T. Gross: *Topology Discovery for Large Ethernet Networks*. In: *Proceedings of the ACM SIGCOMM*, Seiten 237–248, 2001. Auch verfügbar unter `http://www.cs.wm.edu/~lowekamp/papers/lowekamp-topology-sigcomm01.ps` (20.07.2009).

[46] Luckie, M., Y. Hyun und B. Huffaker: *Traceroute Probe Method and Forward IP Path Inference*. In: *Proceedings of the ACM SIGCOMM*, Seiten 311–324, 2008. Auch verfügbar unter `http://doi.acm.org/10.1145/1452520.1452557` (20.07.2009).

[47] Lyon, G.: *Host Discovery*. Verfügbar unter `http://nmap.org/book/man-host-discovery.html` (20.07.2009).

[48] Lyon, G.: *Port Scanning Techniques*. Verfügbar unter `http://nmap.org/book/man-port-scanning-techniques.html` (20.07.2009).

[49] Mao, Y., H. Jamjoom, S. Tao und J. Smith: *NetworkMD: Topology Inference and Failure Diagnosis in the Last Mile*. In: *Proceedings of the ACM SIGCOMM*, Seiten 189–202, 2007. Auch verfügbar unter `http://doi.acm.org/10.1145/1298306.1298333` (20.07.2009).

[50] Maston, M.: *Managing Windows with WMI*, 1999. Verfügbar unter `http://msdn.microsoft.com/en-us/library/ms811533.aspx` (20.07.2009).

[51] McCloghrie, K. und F. Kastenholz: *The Interfaces Group MIB*. RFC 2863, Internet Engineering Task Force, 2000. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc2863.txt` (20.07.2009).

[52] McCloghrie, K. und M. Rose: *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. RFC 1213, Internet Engineering Task Force, 1991. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc1213.txt` (20.07.2009).

[53] Meiss, M. und S. Wallace: *Standards-Based Discovery of Switched Ethernet Topology*. Verfügbar unter `http://steinbeck.ucs.indiana.edu/~mmeiss/papers/Ethernet.pdf` (20.07.2009).

[54] Microsoft Corporation: *NetWkstaUserEnum Function*. Verfügbar unter `http://msdn.microsoft.com/en-us/library/aa370669(VS.85).aspx` (20.07.2009).

[55] Mockapetris, P.: *Domain names - concepts and facilities.* RFC 1034, Internet Engineering Task Force, 1987. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc1034.txt` (20.07.2009).

[56] Mockapetris, P.: *Domain names - implementation and specification.* RFC 1035, Internet Engineering Task Force, 1987. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc1035.txt` (20.07.2009).

[57] Montigny-Leboeuf, A. de und F. Massicotte: *Passive Network Discovery for Real Time Situation Awareness.* In: *RTO-MP-IST-041.* Information Systems Technology Panel, NATO Research and Technology Organisation, 2004. Auch verfügbar unter `http://ftp.iasi.roedu.net/packages/snort/docs/industry/ADeMontigny_NatoISTToulouse2004.pdf` (20.07.2009).

[58] Nazir, F., T. Tarar, F. Javed, H. Suguri, H. Ahmad und A. Ali: *Constella: A Complete IP Network Topology Discovery Solution.* In: *Managing Next Generation Networks and Services*, Lecture Notes in Computer Science, Seiten 425–436. Springer, 2007. Auch verfügbar unter `http://www.springerlink.com/content/g130938r21525844/fulltext.pdf` (20.07.2009).

[59] Network Working Group: *Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods.* RFC 1001, Internet Engineering Task Force, 1987. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc1001.txt` (20.07.2009).

[60] Norseth, K. und E. Bell: *Definitions of Managed Objects for Bridges.* RFC 4188, Internet Engineering Task Force, 2005. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc4188.txt` (20.07.2009).

[61] Plummer, D.: *An Ethernet Address Resolution Protocol.* RFC 826, Internet Engineering Task Force, 1982. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc826.txt` (20.07.2009).

[62] Postel, J.: *User Datagram Protocol.* RFC 768, Internet Engineering Task Force, 1980. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc768.txt` (20.07.2009).

[63] Postel, J.: *Internet Control Message Protocol.* RFC 792, Internet Engineering Task Force, 1981. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc792.txt` (20.07.2009).

[64] Postel, J.: *Internet Protocol.* RFC 791, Internet Engineering Task Force, 1981. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc791.txt` (20.07.2009).

[65] Postel, J.: *Transmission Control Protocol.* RFC 793, Internet Engineering Task Force, 1981. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc793.txt` (20.07.2009).

[66] Postel, J.: *Echo Protocol.* RFC 862, Internet Engineering Task Force, 1983. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc862.txt` (20.07.2009).

[67] Rackspace US Inc.: *Rackspace Virtualization Survey.* Verfügbar unter `http://www.rackspace.com/downloads/surveys/VirtualizationSurvey.pdf` (20.07.2009), 2007.

[68] Research In Motion Limited: *BlackBerry.com.* `http://www.blackberry.com/` (20.07.2009).

[69] Schönwälder, J. und H. Langendörfer: *How to Keep Track of Your Network Configuration.* In: *Proceedings of the USENIX Large Installation System Administration Conference*, 1993. Auch verfügbar unter `ftp://ftp.ibr.cs.tu-bs.de/pub/local/papers/discover.ps.gz` (20.07.2009).

[70] Sherwood, R. und N. Spring: *Touring the Internet in a TCP Sidecar.* In: *Proceedings of the ACM SIGCOMM*, Seiten 339–344, 2006. Auch verfügbar unter `http://doi.acm.org/10.1145/1177080.1177093` (20.07.2009).

[71] Shirai, T., H. Saito und K. Taura: *A Fast Topology Inference: A Building Block for Network-aware Parallel Processing.* In: *Proceedings of the HPDC Symposium*, Seiten 11–22, 2007. Auch verfügbar unter `http://doi.acm.org/10.1145/1272366.1272369` (20.07.2009).

[72] Siamwalla, R., R. Sharma und S. Keshav: *Discovering Internet Topology.* Technischer Bericht, Cornell Network Research Group, Department of Computer Science, Cornell University, Ithaca, 1999. Auch verfügbar unter `http://www.cs.cornell.edu/skeshav/papers/discovery.pdf` (20.07.2009).

[73] Sood, A.: *Detecting VMwares Remotely.* Verfügbar unter `http://www.secniche.org/papers/Detecting_Vmwares_Remotely.pdf` (20.07.2009).

[74] Spring, N., M. Dontcheva, M. Rodrig und D. Wetherall: *How to Resolve IP Aliases.* Technischer Bericht 04-05-04, Department of Computer Science and Engineering, University of Washington, Seattle, 2004. Auch verfügbar unter `http://www.cs.washington.edu/homes/nspring/papers/ally.ps` (20.07.2009).

[75] Srisuresh, P. und K. Egevang: *Traditional IP Network Address Translator (Traditional NAT).* RFC 3022, Internet Engineering Task Force, 2001. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc3022.txt` (20.07.2009).

[76] Stott, D.: *Layer-2 Path Discovery Using Spanning Tree MIBs.* Technischer Bericht, Avaya Labs Research, Avaya Inc., Basking Ridge, 2002. Auch verfügbar unter `http://citeseer.ist.psu.edu/stott02layer.html` (20.07.2009).

[77] Stott, D.: *SNMP-based Layer-3 Path Discovery.* Technischer Bericht, Avaya Labs Research, Avaya Inc., Basking Ridge, 2002. Auch verfügbar unter `http://citeseer.ist.psu.edu/587637.html` (20.07.2009).

[78] Sun, Y., Z. Shi und Z. Wu: *A Discovery Algorithm for Physical Topology in Switched Ethernets.* In: *Proceedings of the IEEE Conference on Local Computer Networks*, Seiten 311–317, 2005. Auch verfügbar unter `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1550871&isnumber=33047` (20.07.2009).

[79] Tanenbaum, A.: *Computer Networks.* Prentice Hall, 4. Auflage, 2002. ISBN 978-0-13-066102-9.

[80] Umamaheswaran, V.: *UTF-EBCDIC.* Unicode Technical Report 16, 2002. Verfügbar unter `http://www.unicode.org/reports/tr16/` (20.07.2009).

[81] UPnP Forum: *UPnP Resources.* Verfügbar unter `http://upnp.org/resources/default.asp` (20.07.2009).

[82] Vigna, G., F. Valeur, J. Zhou und R. Kemmerer: *Composable Tools for Network Discovery and Security Analysis.* In: *Proceedings of the IEEE Annual Computer Security Applications Conference*, Seiten 14–24, 2002. Auch verfügbar unter `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1176274&isnumber=26404` (20.07.2009).

[83] Waddington, D., F. Chang, R. Viswanathan und B. Yao: *Topology Discovery for Public IPv6 Networks.* SIGCOMM Computer Communications Review, 33(3):59–68, 2003. Auch verfügbar unter `http://doi.acm.org/10.1145/956993.957001` (20.07.2009).

[84] Waldbusser, S.: *Remote Network Monitoring Management Information Base Version 2.* RFC 4502, Internet Engineering Task Force, 2006. Auch verfügbar unter `http://tools.ietf.org/rfc/rfc4502.txt` (20.07.2009).

[85] Wei-hua, J., L. Wei-hua und D. Jun: *The Application of ICMP Protocol in Network Scanning.* In: *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies*, Seiten 904–906, 2003. Auch verfügbar unter `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1236446&isnumber=27719` (20.07.2009).

[86] Wood, D., S. Coleman und M. Schwartz: *Fremont: A System for Discovering Network Characteristics and Problems.* In: *Proceedings of the USENIX Winter Conference*, Seiten 335–348, 1993. Auch verfügbar unter `http://www.codeontheroad.com/papers/Fremont.pdf` (20.07.2009).

[87] Wouters, P.: *Designing a Safe Network Using Firewalls.* Linux Journal, Seite 1, 1997. Auch verfügbar unter `http://portal.acm.org/citation.cfm?id=326911.326912` (20.07.2009).

[88] Young, C. und W. Xu: *CISCO-VTP-MIB*, 2004. Verfügbar unter `ftp://ftp.cisco.com/pub/mibs/v2/CISCO-VTP-MIB.my` (20.07.2009).

[89] Zedler, M. und D. Gantenbein: *Physical Location Awareness for Enterprise IT Assets.* In: *Proceedings of the IEEE INFOCOM*, Seiten 1–6, 2008. Auch verfügbar unter `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4544617&isnumber=4544573` (20.07.2009).

# Anhang B

# Network Explorer Handbuch

Dieses Handbuch beschreibt detailliert die Funktion, Installation und Verwendung der *Network Explorer* Applikation und ihrer Komponenten aus Benutzersicht.

## B.1 Beschreibung

Der *Network Explorer* ist ein Java-Programm, mit dem automatisch Informationen über ein Netzwerk gesammelt werden können. Es handelt sich dabei jedoch um kein Hacker-Tool, um ein Netzwerk auszuspionieren, sondern um ein Werkzeug zur Netzwerk-Administration (daher sind auch entsprechende administrative Kenntnisse und Zugänge vonnöten).

Die vom Programm gewonnenen Informationen enthalten unter anderem:

- Im Netzwerk vorhandene Geräte (Workstations, Server, Router, Switches etc.)
- Informationen über diese Geräte (Typ, Adressen, Namen, Anschlussort etc.)
- Layer 3 Topologie (Subnetze, Router, Routen etc.)
- Layer 2 Topologie (VLANs, Switches, Hubs, Verbindungen etc.)

Es werden bei der Analyse des Netzwerks auch Technologien, wie virtuelle Maschinen, virtuelle Router (*VRRP/HSRP*), Link Aggregation, ARP Proxies etc. berücksichtigt. Weiters ist das Programm unabhängig von bestimmten Herstellern oder Typen von Infrastrukturkomponenten und kann auch mit einem heterogenen Umfeld umgehen.

Um die genannten Informationen zu erfassen, greift das Programm direkt auf das Netzwerk zu und sendet einerseits Probe Packets, um nach Geräten

zu suchen, und liest andererseits Daten über SNMP von den Infrastrukturkomponenten aus. Dafür benötigt das Programm lediglich minimale Vorkenntnisse, nämlich die verwendeten SNMP Communities – weitere Daten können optional angegeben werden.

Das Programm bietet mehrere Funktionen. Dazu gehören:

**Explorer:** Sammelt die genannten Daten im Netzwerk.

**Query:** Erlaubt es, die gesammelten Daten tabellarisch abzufragen.

**Mapper:** Kann verschiedene grafische Netzwerk Maps erstellen.

Das Programm hat gewisse Einschränkungen. Dazu gehört, dass es in der derzeitigen Version nur auf *Microsoft Windows* im Single-User-Betrieb einsetzbar ist. Weiters wurde es lediglich für das Durchsuchen und Analysieren von eingegrenzten Local Area Networks – auf denen *Ethernet* und *TCP/IP* (*IPv4*) eingesetzt wird – konzipiert. Das Discovery bietet zudem nur einen Snapshot des Netzwerkes von dem Zeitpunkt, zu dem das Programm ausgeführt wurde, aber keine kontinuierliche Überwachung oder damit zusammenhängende History-Funktionen und dergleichen. Das Discovery ist nur von der Maschine aus möglich, auf der das Programm läuft, und kann dabei nur über jeweils ein Interface dieses Computers geschehen.

## B.2 Voraussetzungen

Damit das Programm erfolgreich eingesetzt werden kann, müssen einige (technische und administrative) Voraussetzungen erfüllt sein:

- Es muss eine *Microsoft Windows*-Plattform für die Installation zur Verfügung stehen.
- Auf dieser Plattform muss *Java* (mindestens in der Version 1.6) installiert sein bzw. installiert werden können.
- Der Benutzer muss sowohl zur Installation, als auch zur Ausführung des Programms über *Administrator-Rechte* auf der Maschine verfügen.
- Idealerweise handelt es sich dabei um eine *physische Maschine* (Tests haben Probleme beim Senden und Empfangen von VLAN getaggten Frames auf *VMware*-Installationen gezeigt, auch wenn diese laut Anleitung korrekt für *Virtual Guest Tagging* konfiguriert waren).
- Auf die Switches und Router im Netzwerk muss über *SNMPv1* zugegriffen werden können.

- Die für diesen Zugriff nötigen *SNMP Read Communities* müssen bekannt sein.
- Das Programm unterstützt einerseits Geräte, welche *standardisierte MIBs* implementieren und andererseits Geräte, welche proprietäre MIBs der Firma *Cisco* anbieten. Proprietäre MIBs anderer Hersteller werden derzeit nicht unterstützt.
- Es sollte auch administrativer Zugang zum Switch, an dem die Discovery-Maschine angeschlossen ist, verfügbar sein, damit dessen Ports für das Senden und Empfangen von VLAN getaggten Frames umkonfiguriert werden können (dies ist jedoch nicht zwingend erforderlich).

## B.3 Installation

Zur Installation müssen folgende Schritte ausgeführt werden (Voraussetzung dafür sind Adminstrator-Rechte auf der Maschine):

1. Entpacken der Datei `Network Explorer.zip` in ein beliebiges Verzeichnis. Alle Programmdateien und dazugehörigen Bibliotheken werden bereits in der passenden bzw. entsprechend adaptierten Form (z.B. für das Senden und Empfangen von VLAN getaggten Frames) ausgeliefert.
2. Installation von *Java 1.6* oder höher (dies kann unter `http://www.java.com/` bezogen werden).
3. Installation von *WinPcap* (dies liegt den Installations-Files bei oder kann unter `http://www.winpcap.org/` bezogen werden).
4. Installation von *Graphviz* (dies liegt den Installations-Files bei oder kann unter `http://www.graphviz.org/` bezogen werden).
5. Falls *MATLAB* bereits auf der selben Maschine installiert ist, kann es u.U. zu Versionskonflikten mit Graphviz kommen. Diese können behoben werden, indem entweder die `PATH` Umgebungsvariable so geändert wird, dass das Graphviz-Verzeichnis vor dem MATLAB-Verzeichnis aufscheint, das MATLAB-Verzeichnis ganz entfernt wird oder indem die Dateien `dot.exe` und `neato.exe` im MATLAB-Verzeichnis gelöscht oder umbenannt werden (z.B. in `dot.exe.bak` und `neato.exe.bak`).
6. Der Computer muss nach der Installation von Graphviz eventuell neu gestartet werden, damit die neuen Einstellungen der `PATH` Umgebungsvariablen wirksam werden.

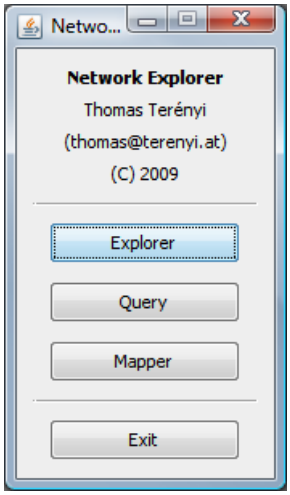


**Abbildung B.1:** Network Explorer Hauptmenü

7. In seltenen Fällen muss die Netzwerkkarte der Discovery-Maschine extra umkonfiguriert werden, sodass VLAN Tags nicht automatisch von der Karte entfernt werden, bevor sie Frames an die Software weitergibt, womit das Senden und Empfangen von VLAN getaggten Frames möglich wird. Nähere Informationen dazu sind der Dokumentation der jeweiligen Netzwerkkarte zu entnehmen.

8. Der Switch Port, an den die Discovery-Maschine angeschlossen wird, ist so umzukonfigurieren, sodass VLAN getaggte Frames in alle VLANs gesendet und aus allen VLANs empfangen werden können. Hierbei wird empfohlen, sich dafür an den zuständigen Netzwerk-Administrator zu wenden. Dieser und der vorige Schritt sind nur notwendig, wenn ein *ARP Discovery* mit VLAN getaggten Frames gewünscht wird.

## B.4 Bedienung

Das Programm wird über Ausführen der Datei `Network Explorer.bat` gestartet (wobei für den direkten Zugriff auf das Netzwerk während des Discoverys Adminstrator-Rechte auf der Maschine benötigt werden). Hierauf öffnet sich das Hauptmenü, in welchem die gewünschte Funktion ausgewählt oder das Programm beendet werden kann.

Fortgeschrittene Anwender können beim Start des Programms zusätzliche Kommandozeilenoptionen angeben (diese sind nicht case-sensitive und können beliebig kombiniert werden):

`-noSNMPDiscovery` Beim Discovery wird das *SNMP Discovery* übersprungen.

`-noVLANDiscovery` Beim Discovery wird das *VLAN Discovery* übersprungen.

`-noARPDiscovery` Beim Discovery wird das *ARP Discovery* übersprungen.

`-noARPTableDiscovery` Beim Discovery wird das *ARP Table Discovery* übersprungen.

`-noICMPDiscovery` Beim Discovery wird das *ICMP Discovery* übersprungen.

`-noDNSDiscovery` Beim Discovery wird das *DNS Discovery* übersprungen.

`-noSTPDiscovery` Beim Discovery wird das *STP Discovery* übersprungen.

`-noAFTDiscovery` Beim Discovery wird das *AFT Discovery* übersprungen.

`-noVMDiscovery` Beim Discovery wird die Erkennung von virtuellen Maschinen übersprungen.

`-noARPProxyDiscovery` Beim Discovery wird die Erkennung von ARP Proxies übersprungen.

`-noVirtualRouterDiscovery` Beim Discovery wird die Erkennung von virtuellen Routern (VRRP/HSRP) übersprungen.

`-noVirtualRoutingEngineDiscovery` Beim Discovery wird die Erkennung von virtuellen Routing Engines (auf einem physischen Switch) übersprungen.

`-noNonSNMPSwitchDiscovery` Beim Discovery wird die Erkennung von Nicht-SNMP-Switches übersprungen.

`-noTopologyInference` Beim Discovery wird die Berechnung der Topologie übersprungen.

`-dontUpdateSubnetMasks` Beim Discovery werden die IP-Adressen nicht dem jeweils kleinsten passenden Subnetz zugeordnet.

`-discoverActiveNamesOnly` Das DNS Discovery wird nur für jene IP-Adressen durchgeführt, hinter denen sich auch ein aktives Gerät verbirgt (anstatt für alle IP-Adressen im Bereich).

`-arpProxyIPCount=`*`n`* Setzt die minimale Anzahl von IP-Adressen, die einer MAC-Adresse zugeordnet sein müssen, damit das dazugehörige Gerät als ARP Proxy erkannt wird.

`-ignoreVLAN=`*`id`* Das VLAN mit der gegebenen ID wird beim Discovery ignoriert. Diese Option kann auch mehrfach vorkommen.

`-ignoreIP=`*`ip`* Die gegebene IP-Adresse wird beim Discovery ignoriert. Diese Option kann auch mehrfach vorkommen.

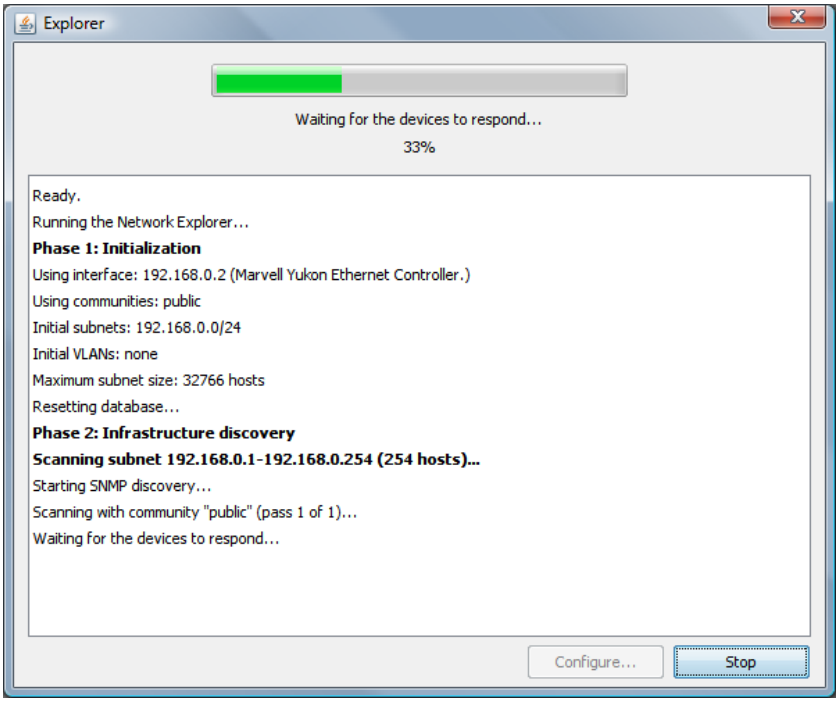


**Abbildung B.2:** Explorer Window

### B.4.1 Explorer

Die *Explorer*-Funktion erlaubt es, Daten über das Netzwerk zu sammeln. Der Vorgang wird durch den `Start/Stop` Button gestartet bzw. abgebrochen. Während der Explorer läuft, zeigt ein Progress Bar den Fortschritt bei der aktuellen Tätigkeit an. Im Log-Bereich werden Statusmeldungen ausgegeben, welche entsprechend ihres Typs eingefärbt sind (Debug-Nachrichten grau, Informationen schwarz, Warnungen gelb, Fehler orange und fatale Fehler rot). Die Nachrichten lassen sich mittels `Strg+C` auch in ein anderes Fenster kopieren.

Sobald ein neuer Durchlauf beginnt, wird der Inhalt der Datenbank gelöscht und die neuen Daten erst dann gespeichert, wenn der Vorgang komplett abgeschlossen ist. Wird der Vorgang abgebrochen, bleibt die Datenbank leer. Nach einem erfolgreichen Durchlauf kann das Fenster über den `X` Button geschlossen und die Daten mit der *Query*- oder *Mapper*-Funktion betrachtet werden.

Beim Discovery werden folgende Phasen durchlaufen:

1. *Initialization*: Variablen werden initialisiert und die Datenbank wird zurückgesetzt.

2. *Infrastructure Discovery*: Informationen über die Netzwerkinfrastruktur (Router, Switches, Subnetze, VLANs etc.) werden erhoben.

   (a) *SNMP Discovery*: Es wird in jedem Subnetz nach SNMP-Geräten gesucht und von den gefundenen Geräten Informationen (z.B. Interfaces, Routing Tables etc.) ausgelesen. Dabei werden auch neue Subnetze erkannt.

   (b) *VLAN Discovery*: Die definierten VLANs werden von allen Switches ausgelesen.

3. *Host Discovery*: Geräte werden im Netzwerk gesucht.

   (a) *ARP Table Discovery*: In jedem Subnetz werden die ARP Tables der dazugehörigen Router (ev. in mehreren Durchgängen) ausgelesen. Vor jedem Auslesen werden die entsprechenden IP-Adressen im jeweiligen Subnetz mit ICMP gepingt.

   (b) *ARP Discovery*: Mittels ARP wird nach aktiven Geräten gesucht, wobei VLAN getaggte und normale Frames versendet werden (ein Durchlauf pro VLAN).

   (c) *ICMP Discovery*: Mittels ICMP wird nach aktiven Geräten gesucht.

   (d) *DNS Discovery*: Mittels DNS wird nach einem Namen zu jeder IP-Adresse gesucht.

4. *Topology Discovery*: Die Topologie wird bestimmt.

   (a) *STP Discovery*: Die STP-Informationen werden von den Switches ausgelesen und der Spanning Tree wird aufgebaut.

   (b) *AFT Discovery*: Die AFTs der Switches werden (ev. in mehreren Durchgängen) ausgelesen. Vor jedem Auslesen werden die entsprechenden Geräte sowohl mit ARP, als auch mit ICMP gepingt.

   (c) *Topology Inference*: Die Topologie wird errechnet. Zusätzlich werden virtuelle Maschinen, virtuelle Router, virtuelle Routing Engines, ARP Proxies und Nicht-SNMP-Switches identifiziert.

5. *Finalization*: Die gesammelten Daten werden in die Datenbank geschrieben.

Bevor der Explorer zum ersten Mal gestartet werden kann, muss er konfiguriert werden (ansonsten bleibt der `Start` Button deaktiviert). Um dies zu tun oder auch um später die Einstellungen zu ändern, kann das *Settings*-Fenster über den `Configure...` Button geöffnet werden.

Folgende Einstellungen können vorgenommen werden:

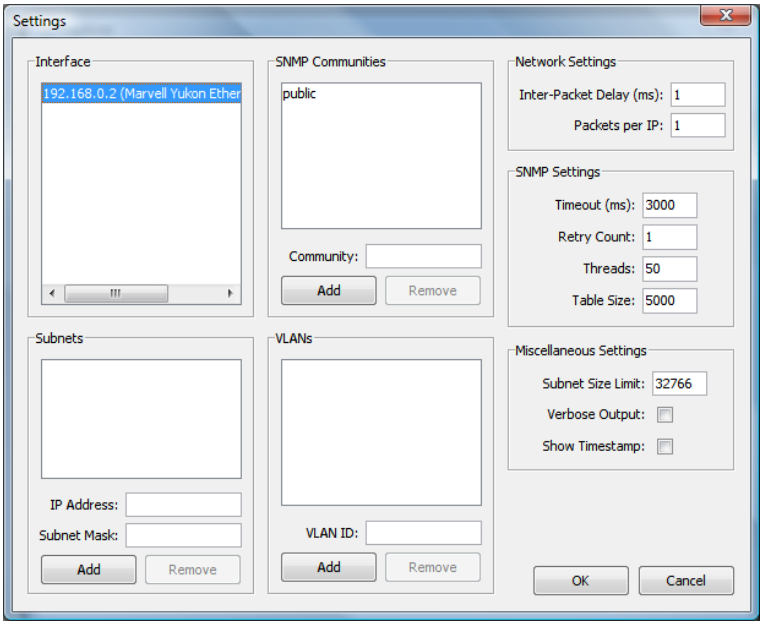


**Abbildung B.3:** Settings Window

`Interface` Das Interface der Maschine, welches zum Scannen des Netzwerkes verwendet werden soll. Genau eines muss ausgewählt sein.

`SNMP Communities` Die SNMP Read Communities, die zum Suchen nach SNMP-Geräten bzw. zum Auslesen von Daten verwendet werden sollen. Mindestens eine muss angegeben werden.

`Subnets` Eine optionale explizite Angabe von Subnetzen, die durchsucht werden sollen.

`VLANs` Eine optionale explizite Angabe von VLANs, die durchsucht werden sollen.

`Inter-Packet Delay` Die Zeit in Millisekunden, die zwischen dem Senden zweier Pakete bei den Scans gewartet werden soll. Dies dient dazu, um das Netzwerk nicht mit zu vielen Paketen in einer zu kurzen Zeit zu überfluten.

`Packets per IP` Die Anzahl an Durchläufen bei jedem Scan-Vorgang. Dies dient dazu, um mehrere Pakete an jede IP-Adresse zu senden, falls damit gerechnet wird, dass Pakete verloren gehen könnten.

`Timeout` Das Timeout in Millisekunden für SNMP-Operationen.

`Retry Count` Die Anzahl, wie oft durch ein Timeout fehlgeschlagene SNMP-Operationen wiederholt werden sollen.

`Threads` Die Anzahl an Threads, die verwendet werden, um gleichzeitig Daten über SNMP von verschiedenen Geräten auszulesen. Da SNMP-Operationen einerseits relativ langsam sind, andererseits aber wenig Netzwerklast verursachen, kann mit mehreren SNMP-Geräten parallel kommuniziert werden. Diese Einstellung gilt analog auch für das *DNS Discovery*, wo ebenfalls mit mehreren Threads gearbeitet wird.

`Table Size` Die angenommene maximale Anzahl von Zeilen in ARP Tables und AFTs. Es werden (falls nötig) in mehreren Durchgängen jeweils diese Anzahl von Geräten gepingt und danach die Tabellen ausgelesen. Dies dient dazu, um zu verhindern, dass Daten nicht gefunden werden, weil entweder Switches oder Router nur eine gewisse Anzahl von Zeilen in ihren Tabellen speichern können oder weil die Ausleseoperationen so lange dauern, dass in der Zwischenzeit Einträge durch ein Timeout entfernt wurden.

`Subnet Size Limit` Die maximale Größe (Anzahl an Hosts) von Subnetzen, die vom Programm gescannt werden. Größere Subnetze werden ignoriert. Dies dient dazu, um zu große Subnetze auszuschließen bzw. den Scan auf das lokale Netzwerk zu beschränken.

`Verbose Output` Gibt an, ob detaillierte Debug-Nachrichten im Log ausgegeben werden sollen.

`Show Timestamp` Gibt an, ob alle Log-Meldungen mit einem Zeitstempel versehen werden sollen.

Durch einen Klick auf `OK` werden die neuen Einstellungen übernommen, durch einen Klick auf `Cancel` verworfen.

### B.4.2 Query

Mit der *Query*-Funktion können Daten abgefragt und tabellarisch dargestellt werden. Voraussetzung dafür ist, dass auch Daten vorhanden sind (der Explorer also erfolgreich durchgelaufen ist).

Im Query-Bereich kann eine *SQL*-Abfrage (`SELECT`) eingegeben und durch Betätigen von `Execute` ausgeführt werden, worauf das Ergebnis in der Tabelle dargestellt wird.

In dieser Tabelle können Spalten durch Ziehen des Spaltenkopfs umgereiht oder Zeilen durch Klicken auf einen Spaltenkopf sortiert werden (mehrmaliges Klicken auf den selben Spaltenkopf ändert die Sortierrichtung).

Weiters kann die Breite von Spalten durch das Ziehen an der Trennlinie zwischen den Spaltenköpfen verändert werden. Unter der Tabelle wird die

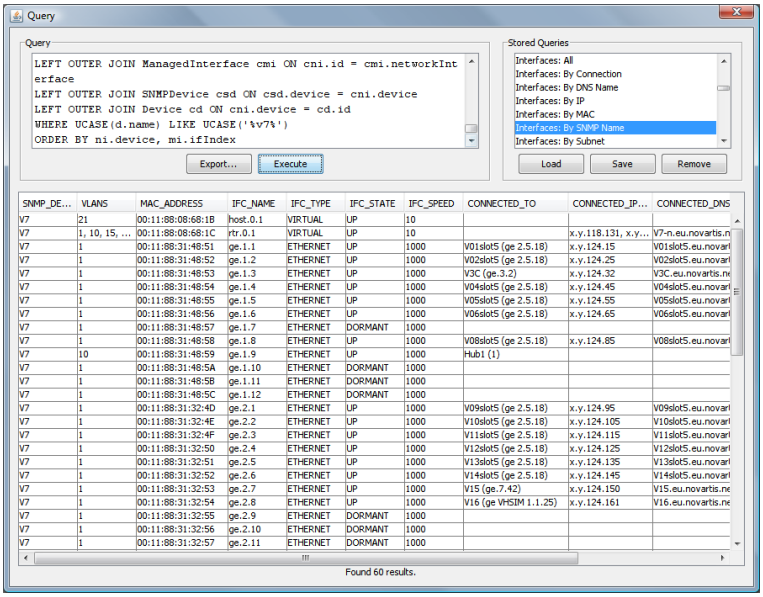


**Abbildung B.4:** Query Window

Anzahl der gefundenen Ergebnisse sowie (falls anwendbar) die Anzahl der markierten Zeilen angezeigt. Bleibt der Mouse-Cursor länger über einer Zelle, so wird deren Inhalt in einem Tooltip angezeigt.

Der Inhalt von markierten Zellen kann mittels `Strg+C` in andere Fenster kopiert werden. Soll der Inhalt der gesamten Tabelle exportiert werden, bietet sich alternativ dazu die *Export*-Funktion an. Durch Klicken auf `Export...` und Auswahl einer Zieldatei wird in diese der Inhalt der Tabelle im *CSV*-Format abgespeichert.

Soll der Inhalt einer Tabelle gedruckt werden, so bietet sich an, diesen entweder direkt in ein anderes Programm (z.B. *Word*) zu kopieren und das entsprechende Dokument auszudrucken oder die Tabelle zuerst zu exportieren und dann das exportierte File (z.B. in *Excel*) zu öffnen und dort auszudrucken.

Es können nicht nur SQL-Abfragen mit der Query-Funktion sondern auch SQL-Kommandos (`INSERT`, `UPDATE` und `DELETE`) ausgeführt werden, falls die gesammelten Daten angepasst werden sollen.

Damit häufig verwendete Abfragen nicht immer neu geschrieben werden müssen, erlaubt es die Funktion *Stored Queries* diese für eine spätere Verwendung zu speichern. Mittels `Save` wird das aktuelle Query gespeichert, `Remove` entfernt das markierte Query und `Load` lädt die selektierte Abfrage und führt sie auch gleich aus.

Zur einfacheren Handhabung sind etwa 120 Abfragen bereits vordefiniert. Folgende sind verfügbar:

`Devices: All` Listet alle gefundenen Geräte auf.

`Devices: By Connection` Sucht Geräte, die an der gegebenen Stelle angeschlossen sind.

`Devices: By DNS Name` Sucht Geräte, die den gegebenen Netzwerknamen haben.

`Devices: By IP` Sucht Geräte, die die gegebene IP-Adresse haben.

`Devices: By MAC` Sucht Geräte, die die gegebene MAC-Adresse haben.

`Devices: By SNMP Name` Sucht Geräte, die den gegebenen SNMP-Namen haben.

`Devices: By Subnet` Sucht Geräte, die zum gegebenen Subnetz gehören.

`Devices: By VLAN` Sucht Geräte, die sich im gegebenen VLAN befinden.

`Devices: Multiple IPs` Listet alle Geräte auf, die mehrere IP-Adressen haben.

`Devices: Multiple MACs` Listet alle Geräte auf, die mehrere MAC-Adressen haben.

`Devices: Multiple Names` Listet alle Geräte auf, die mehrere Netzwerknamen haben.

`Devices: Multiple Subnets` Listet alle Geräte auf, die zu mehreren Subnetzen gehören.

`Devices: Multiple VLANs` Listet alle Geräte auf, die sich in mehreren VLANs befinden.

`Devices: No IPs` Listet alle Geräte auf, die keine IP-Adresse haben.

`Devices: No MACs` Listet alle Geräte auf, zu denen keine MAC-Adresse bestimmt werden konnte.

`Devices: No Names` Listet alle Geräte auf, die keinen Netzwerknamen haben.

`Devices: No VLANs` Listet alle Geräte auf, bei denen das VLAN nicht bestimmt werden konnte.

`Found with: AFT Discovery` Listet alle Geräte auf, die vom *AFT Discovery* gefunden wurden.

`Found with: AFT Discovery Only` Listet alle Geräte auf, die nur vom *AFT Discovery* gefunden wurden.

`Found with: ARP Discovery` Listet alle Geräte auf, die vom *ARP Discovery* gefunden wurden.

`Found with: ARP Discovery Only` Listet alle Geräte auf, die nur vom *ARP Discovery* gefunden wurden.

`Found with: ARP Table Discovery` Listet alle Geräte auf, die vom *ARP Table Discovery* gefunden wurden.

`Found with: ARP Table Discovery Only` Listet alle Geräte auf, die nur vom *ARP Table Discovery* gefunden wurden.

`Found with: DNS Discovery` Listet alle Geräte auf, die vom *DNS Discovery* gefunden wurden.

`Found with: DNS Discovery Only` Listet alle Geräte auf, die nur vom *DNS Discovery* gefunden wurden.

`Found with: ICMP Discovery` Listet alle Geräte auf, die vom *ICMP Discovery* gefunden wurden.

`Found with: ICMP Discovery Only` Listet alle Geräte auf, die nur vom *ICMP Discovery* gefunden wurden.

`Found with: Route Discovery` Listet alle Geräte auf, die in Routen gefunden wurden.

`Found with: Route Discovery Only` Listet alle Geräte auf, die nur in Routen gefunden wurden.

`Found with: SNMP Discovery` Listet alle Geräte auf, die vom *SNMP Discovery* gefunden wurden.

`Found with: SNMP Discovery Only` Listet alle Geräte auf, die nur vom *SNMP Discovery* gefunden wurden.

`Hubs: All` Listet alle Hubs auf.

`Hubs: By Name` Listet alle Ports des gegebenen Hubs auf.

`Interfaces: All` Listet alle gefundenen Interfaces auf.

`Interfaces: By Connection` Sucht Interfaces, die an der gegebenen Stelle angeschlossen sind.

`Interfaces: By DNS Name` Listet alle Interfaces auf, die zu dem Gerät mit dem gegebenen Netzwerknamen gehören.

`Interfaces: By IP` Listet alle Interfaces auf, die zu dem Gerät mit der gegebenen IP-Adresse gehören.

`Interfaces: By MAC` Sucht Interfaces, die die gegebene MAC-Adresse haben.

`Interfaces: By SNMP Name` Listet alle Interfaces auf, die zu dem Gerät mit dem gegebenen SNMP-Namen gehören.

`Interfaces: By Subnet` Listet alle Interfaces auf, die zu einem Gerät im gegebenen Subnetz gehören.

`Interfaces: By VLAN` Sucht Interfaces, die sich im gegebenen VLAN befinden.

`Interfaces: Connected` Listet alle Interfaces auf, die verbunden sind.

`Interfaces: Disconnected` Listet alle Interfaces auf, die nicht verbunden sind.

`Interfaces: Duplicate MACs` Listet alle MAC-Adressen auf, die mehrfach vergeben sind.

`Interfaces: Multiple IPs` Listet alle Interfaces auf, die mit mehreren IP-Adressen assoziiert sind.

`Interfaces: Multiple VLANs` Listet alle Interfaces auf, die sich in mehreren VLANs befinden.

`Interfaces: No IPs` Listet alle Interfaces auf, die mit keiner IP-Adresse assoziiert sind.

`Interfaces: No MAC` Listet alle Interfaces auf, bei denen die MAC-Adresse nicht bestimmt werden konnte.

`Interfaces: No VLANs` Listet alle Interfaces auf, bei denen das VLAN nicht bestimmt werden konnte.

`IP Addresses: All` Listet alle gefundenen IP-Adressen auf.

`IP Addresses: By DNS Name` Sucht IP-Adressen, die mit dem gegebenen Netzwerknamen assoziiert sind.

`IP Addresses: By IP` Sucht nach der gegebenen IP-Adresse.

`IP Addresses: By MAC` Sucht IP-Adressen, die mit der gegebenen MAC-Adresse assoziiert sind.

`IP Addresses: By SNMP Name` Sucht IP-Adressen, die zu einem Gerät mit dem gegebenen SNMP-Namen gehören.

`IP Addresses: By Subnet` Listet alle IP-Adressen im gegebenen Subnetz auf.

`IP Addresses: By VLAN` Sucht IP-Adressen, die zu Geräten im gegebenen VLAN gehören.

`IP Addresses: Multiple Interfaces` Listet alle IP-Adressen auf, die mit mehreren Interfaces assoziiert sind.

`IP Addresses: Multiple MACs` Listet alle IP-Adressen auf, die mit mehreren MAC-Adressen assoziiert sind.

`IP Addresses: No Interfaces` Listet alle IP-Adressen auf, die mit keinem Interface assoziiert sind.

`IP Addresses: No MACs` Listet alle IP-Adressen auf, die mit keiner MAC-Adresse assoziiert sind.

`IP Addresses: No Names` Listet alle IP-Adressen auf, die mit keinem Netzwerknamen assoziiert sind.

`Network Names: All` Listet alle gefundenen Netzwerknamen auf.

`Network Names: Multiple IPs` Listet alle Netzwerknamen auf, die mehr als einer IP-Adresse zugeordnet sind.

`Routes: Direct` Listet alle gefundenen direkten Routen auf.

`Routes: Indirect` Listet alle gefundenen indirekten Routen auf.

`SNMP Devices: All` Listet alle gefundenen SNMP-Geräte auf.

`SNMP Devices: By DNS Name` Sucht Geräte, die den gegebenen Netzwerknamen haben.

`SNMP Devices: By IP` Sucht SNMP-Geräte, die die gegebene IP-Adresse haben.

`SNMP Devices: By MAC` Sucht SNMP-Geräte, die die gegebene MAC-Adresse haben.

`SNMP Devices: By Management IP` Sucht SNMP-Geräte, die unter der gegebenen IP-Adresse ansprechbar sind.

`SNMP Devices: By SNMP Community` Sucht SNMP-Geräte, die unter der gegebenen SNMP Community ansprechbar sind.

`SNMP Devices: By SNMP Name` Sucht SNMP-Geräte, die den gegebenen SNMP-Namen haben.

`SNMP Devices: By Subnet` Sucht SNMP-Geräte, die zum gegebenen Subnetz gehören.

`SNMP Devices: By VLAN` Sucht SNMP-Geräte, die sich im gegebenen VLAN befinden.

`SNMP Devices: Miscellaneous` Listet alle diversen SNMP-Geräte auf.

`SNMP Devices: Printers` Listet alle SNMP-fähigen Drucker auf.

`SNMP Devices: Routers` Listet alle SNMP-fähigen Router auf.

`SNMP Devices: Switches (All)` Listet alle SNMP-fähigen Switches auf.

`SNMP Devices: Switches (Cisco VTP)` Listet alle SNMP-fähigen Switches auf, die das *Cisco VTP* unterstützen.

`SNMP Devices: Switches (Q-Bridge)` Listet alle SNMP-fähigen Switches auf, die die *Q-BRIDGE-MIB* unterstützen.

`SNMP Devices: Virtual Machines` Listet alle SNMP-fähigen virtuellen Maschinen auf.

`SNMP Devices: WAN Gateways` Listet alle SNMP-fähigen WAN Gateways auf.

`SNMP Devices: WLAN Access Points` Listet alle SNMP-fähigen WLAN Access Points auf.

`Spanning Tree: All Ports` Listet alle Switch Ports mit STP Informationen auf.

`Spanning Tree: All Ports (Raw)` Listet alle Switch Ports mit STP Informationen auf (und zeigt dabei die Rohdaten an).

`Spanning Tree: Blocking Ports` Listet alle Blocking Switch Ports auf.

`Spanning Tree: Blocking Ports (Raw)` Listet alle Blocking Switch Ports auf (und zeigt dabei die Rohdaten an).

`Spanning Tree: By SNMP Name` Listet alle Ports des gegebenen Switches mit STP Informationen auf.

`Spanning Tree: Forwarding Ports` Listet alle Forwarding Switch Ports auf.

`Spanning Tree: Forwarding Ports (Raw)` Listet alle Forwarding Switch Ports auf (und zeigt dabei die Rohdaten an).

`Spanning Tree: Root Switches` Listet alle Switches auf, die der Root in einem Spanning Tree sind.

`Spanning Tree: Trunk Ports` Listet alle STP Trunk Ports auf.

`Spanning Tree: Trunk Ports (Raw)` Listet alle STP Trunk Ports auf (und zeigt dabei die Rohdaten an).

`Special Devices: ARP Proxies` Listet alle ARP Proxies auf.

`Special Devices: Multi-Homed` Listet alle Geräte mit mehreren Interfaces auf.

`Special Devices: Non-SNMP` Listet alle Geräte auf, die nicht SNMP-fähig sind.

`Special Devices: Printers` Listet alle Drucker auf.

`Special Devices: Routers` Listet alle Router auf.

`Special Devices: SNMP` Listet alle SNMP-fähigen Geräte auf.

`Special Devices: Switches (Non-SNMP)` Listet alle Switches auf, die nicht SNMP-fähig sind.

`Special Devices: Switches (SNMP)` Listet alle SNMP-fähigen Switches auf.

`Special Devices: Virtual Machines` Listet alle virtuellen Maschinen auf.

`Special Devices: Virtual Routers` Listet alle virtuellen Router (*VRRP*/*HSRP*) auf.

`Special Devices: Virtual Routing Engines` Listet alle virtuellen Routing Engines (auf physischen Switches) auf.

`Special Devices: WAN Gateways` Listet alle WAN Gateways auf.

`Special Devices: WLAN Access Points` Listet alle WLAN Access Points auf.

`Special Interfaces: Link Aggregation` Listet alle Link Aggregation Interfaces auf.

`Special Interfaces: Unknown Type` Listet alle Interfaces auf, deren Typ unbekannt ist.

`Special Interfaces: Uplink` Listet alle Switch Ports auf, die mit einem anderen Switch verbunden sind.

`Special Interfaces: WLAN` Listet alle Funk-Interfaces auf.

`Statistics: Devices` Zeigt, wie viele Geräte welchen Typs gefunden wurden.

`Statistics: Discovery Method` Zeigt, wie viele Geräte mit welcher Methode gefunden wurden.

`Statistics: Interfaces` Zeigt, wie viele Interfaces welchen Zustand haben.

`Statistics: SNMP Communities` Zeigt, wie viele Geräte unter welcher SNMP Community ansprechbar sind.

`Statistics: Switch Ports (by Switch)` Liefert eine Übersicht über den Zustand der Ports eines jeden Switches.

`Statistics: Switch Ports (Overall)` Zeigt, wie viele Switch Ports insgesamt welchen Zustand haben.

`Subnets` Listet alle gefundenen Subnetze auf.

`VLANs: All` Listet alle gefundenen VLANs auf.

`VLANs: By Name` Zeigt, welches VLAN mit welchen Namen auf welchen Switches definiert ist.

`VLANs: By Switch` Listet alle VLANs auf, die auf dem gegebenen Switch definiert sind.

`VLANs: No Devices` Listet alle VLANs auf, in denen sich keine Endgeräte befinden.

Bei den vordefinierten Queries, die einen Suchbegriff erfordern (z.B. eine bestimmte IP-Adresse), öffnet sich ein Fenster, in das dieser (nicht case-sensitive) eingegeben werden kann. Dies erspart das händische Einfügen des Suchbegriffs in das SQL-Query. Standardmäßig wird so gesucht, dass auch eine teilweise Übereinstimmung mit dem Suchbegriff zu einem Treffer führt.

### B.4.3 Mapper

Die *Mapper*-Funktion erlaubt es, die gesammelten Daten grafisch in Form von Netzwerk-Maps darzustellen (daher ist auch hier die Voraussetzung, dass der Explorer erfolgreich durchgelaufen ist). Die Maps verwenden dabei die gängigen Symbole für Netzwerkkomponenten (virtuelle Komponenten werden transparent abgebildet). Nach Auswahl des Typs der Map und Setzen der Optionen, kann die Map durch Klicken auf `Generate` erzeugt werden.

Folgende Arten von Maps können erzeugt werden:

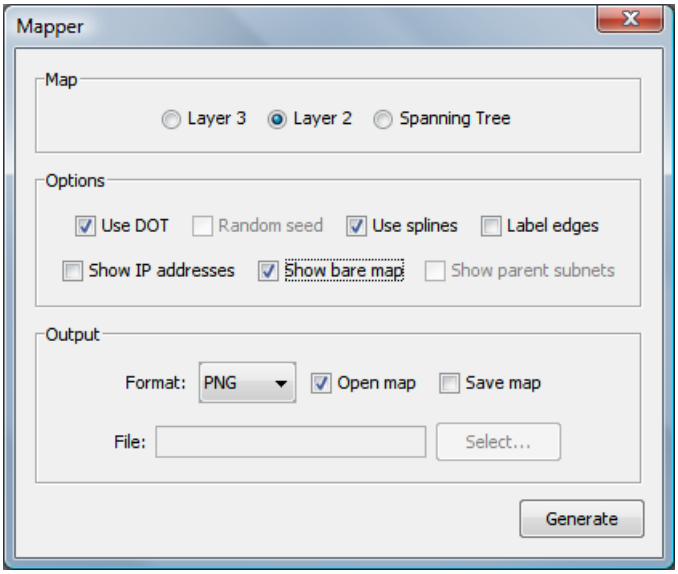


**Abbildung B.5:** Mapper Window

`Layer 3` Diese Map stellt die Layer 3 Topologie des Netzwerkes dar. Eingezeichnet werden Subnetze, Router, Gateways, WLAN Access Points sowie Geräte, die Interfaces in mehreren Subnetzen besitzen. Wird eine *bare Map* erzeugt, so werden nur Subnetze, Router und Gateways abgebildet. In dieser Map stellen dünne Linien Interfaces in dem entsprechenden Subnetz, dicke Linien zusätzlich direkte Routen in ein Subnetz und strichlierte Linien indirekte Routen dar.

`Layer 2` Diese Map stellt die Layer 2 Topologie des Netzwerkes dar. Eingezeichnet werden Switches, Router, Gateways, WLAN Access Points, Hubs sowie Geräte, die mehrfach angeschlossen sind. Wird eine *bare Map* erzeugt, so werden nur Switches, Router, Gateways und WLAN Access Points abgebildet. In dieser Map stellen durchgezogene Linien aktive Verbindungen zwischen je zwei Interfaces und strichlierte Linien durch STP deaktivierte Verbindungen dar. Die Bezeichnung von Root Switches in einem Spanning Tree ist rot eingefärbt.

`Spanning Tree` Diese Map stellt den Spanning Tree, wie er durch STP definiert ist, dar. Eingezeichnet werden nur Switches. Durchgezogene Linien bilden aktive, strichlierte Linien deaktivierte Verbindungen ab. Hierbei stellen die Verbindungen nicht den physischen Zusammenschluss der Switches dar, sondern nur deren Hierarchie im Sinne des STP. Die Bezeichnung von Root Switches in einem Spanning Tree ist rot eingefärbt.

Folgende Optionen stehen bei der Erzeugung von Maps zur Verfügung:

`Use DOT` Gibt an, ob `dot` (anstatt `neato`) zur Erzeugung der Map verwendet werden soll. Das Programm `dot` erzeugt ein hierarchisches Layout, während `neato` ein Netzwerk-Layout generiert. Diese Option gilt nur für Layer 3 und Layer 2 Maps.

`Random seed` Falls `neato` und nicht `dot` zur Erzeugung der Map verwendet wird, so gibt diese Option an, ob zur Generierung des Layouts ein zufälliger Startwert verwendet werden soll. Ist dies der Fall, wird bei jedem Aufruf ein anderes Layout erzeugt. Dadurch können verschiedene Varianten einer Map angefertigt werden.

`Use splines` Gibt an, ob die Verbindungslinien geschwungen oder gerade gezeichnet werden sollen.

`Label edges` Gibt an, ob die Verbindungslinien beschriftet werden sollen. In der Layer 3 Map werden die Kanten mit der IP-Adresse des Interfaces im jeweiligen Subnetz gekennzeichnet, während bei der Layer 2 und der Spanning Tree Map die Bezeichnungen der beiden an der Verbindung beteiligten Interfaces eingezeichnet wird.

`Show IP addresses` Gibt an, ob zusätzlich zum Namen der dargestellten Geräte auch deren IP-Adresse eingezeichnet werden soll.

`Show bare map` Gibt an, ob eine *bare Map*, d.h. eine einfachere Map mit weniger dargestellten Informationen, erzeugt werden soll. Diese Option gilt nur für Layer 3 und Layer 2 Maps.

`Show parent subnets` Gibt ab, ob bei Layer 3 Maps die Hierarchie zwischen Subnetzen (falls es überlappende Subnetze gibt) eingezeichnet werden soll.

Folgende Ausgabe-Optionen gibt es:

`Format` Gibt das Format der Ausgabedatei an.

`Open map` Gibt an, ob die erzeugte Map nach der Generierung geöffnet und angezeigt werden soll.

`Save map` Gibt an, ob die erzeugte Map nach der Generierung in einer Datei gespeichert werden soll.

`File` Gibt die Datei an, in welcher die erzeugte Map nach der Generierung gespeichert werden soll. Diese Datei kann über den `Select...` Button ausgewählt werden.

Soll die erzeugte Map nachträglich editiert werden, so gibt es mehrere Möglichkeiten. Einerseits kann die Map im `SVG`-Format erzeugt werden,

wodurch eine Vektorgrafik generiert wird, die mit einem entsprechenden Zeichenprogramm geöffnet und bearbeitet werden kann. Andererseits kann auch eine Datei im `DOT`- oder `CANON`-Format erzeugt werden, welche nur eine Beschreibung das Graphs im *Graphviz*-Format enthält (beim `DOT`-Format sind dabei schon Layout-Informationen vorhanden, beim `CANON`-Format nicht). Diese Datei kann nun zum Beispiel entweder mit dem Programm `dotty` geöffnet und editiert werden oder die Datei kann als Eingabe für `dot` oder `neato` zur Erzeugung einer Grafik verwendet werden, wobei der Benutzer selbst Optionen nach seinen Wünschen angeben kann.