



Georg-August-Universität  
Göttingen  
Zentrum für Informatik

ISSN 1612-6793  
Nummer ZFI-BM-2003-09

## **Bachelorarbeit**

im Studiengang "Angewandte Informatik"

# **Ein datenbankgestütztes, SNMP-basiertes Netzwerk Informations- und Überwachungssystem**

Daniel van Ross

am

Mathematischen Institut

Bachelor- und Masterarbeiten  
des Zentrums für Informatik  
an der Georg-August-Universität Göttingen

20. August 2003

Georg-August-Universität Göttingen  
Zentrum für Informatik

Lotzestraße 16-18  
37083 Göttingen  
Germany

Tel. +49 (5 51) 39-1 44 02

Fax +49 (5 51) 39-1 44 03

Email [office@informatik.uni-goettingen.de](mailto:office@informatik.uni-goettingen.de)

WWW [www.informatik.uni-goettingen.de](http://www.informatik.uni-goettingen.de)

---

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Göttingen, den 20. August 2003



# Inhaltsverzeichnis

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Einleitung</b>  | <b>5</b>  |
| <b>2</b> | <b>Netzwerküberwachung und -information</b>  | <b>7</b>  |
| 2.1      | Netzwerküberwachung . . . . .  | 7         |
| 2.2      | Netzwerkinformation . . . . .  | 8         |
| 2.2.1    | Informationen für die Administratoren . . . . .                                      | 8         |
| 2.2.2    | Informationen für Benutzer . . . . .   | 9         |
| <b>3</b> | <b>Das Simple Network Management Protocol</b>  | <b>10</b> |
| 3.1      | Was ist SNMP? . . . . .  | 10        |
| 3.2      | Verschiedene SNMP Versionen und RFCs . . . . .                                       | 10        |
| 3.3      | Agenten, Manager und deren Kommunikation . . . . .                                   | 12        |
| 3.4      | Position von SNMP in der TCP/IP-Suite und die daraus resultierenden Folgen . . . . . | 13        |
| 3.5      | Verwaltete Informationen . . . . .   | 14        |
| 3.5.1    | Der SNMP Namensraum . . . . .  | 14        |
| 3.5.2    | Definition von Objekten . . . . .  | 15        |
| 3.5.3    | Erweiterungen durch SMIV2 . . . . .  | 20        |
| <b>4</b> | <b>Überblick über Implementation und Komponenten</b>                                 | <b>22</b> |
| 4.1      | Überblick über die Implementation . . . . .  | 22        |
| 4.2      | Die Komponenten . . . . .  | 22        |
| 4.2.1    | Implementation in Perl . . . . .   | 22        |

|          |  |           |
|----------|--|-----------|
| 4.2.2    | NetSNMP . . . . .  | 23        |
| 4.2.3    | MySQL . . . . .  | 23        |
| 4.2.4    | PHP . . . . .  | 23        |
| 4.2.5    | Präsentation der gesammelten Daten . . . . .                         | 24        |
| <b>5</b> | <b>Die Implementation im Detail</b>                                  | <b>25</b> |
| 5.1      | Testmethoden . . . . .   | 25        |
| 5.1.1    | „Reale Tests“ vs. SNMP-Queries . . . . .                             | 25        |
| 5.1.2    | SNMP basierte Tests . . . . .  | 25        |
| 5.2      | Skripte zur Ergänzung von NetSNMP . . . . .                          | 26        |
| 5.3      | Konfiguration von NetSNMP . . . . .                                  | 26        |
| 5.4      | Implementation des Perl-Teils . . . . .                              | 30        |
| 5.4.1    | Die Konfigurationsdateien . . . . .                                  | 30        |
| 5.4.2    | Durchführen der Tests . . . . .                                      | 32        |
| 5.4.3    | Die Ergebnisse . . . . .   | 32        |
| 5.4.4    | Aufbereitung der Ergebnisse . . . . .                                | 33        |
| 5.4.5    | Sicherung der Ergebnisse . . . . .                                   | 35        |
| 5.4.6    | Wiederholung . . . . .   | 35        |
| 5.5      | Die Tests . . . . .  | 35        |
| 5.5.1    | Erreichbarkeit (conn) . . . . .                                      | 35        |
| 5.5.2    | Webserver (http) . . . . .   | 36        |
| 5.5.3    | FTP Server (ftp), SMTP Server (smtp),<br>POP3 Server (pop) . . . . . | 36        |
| 5.5.4    | Domain Name Service (dns) . . . . .                                  | 36        |
| 5.5.5    | Secure Shell (ssh) . . . . .   | 36        |
| 5.5.6    | SNMP-Agent (snmp) . . . . .  | 36        |
| 5.5.7    | Swapspace (swap) . . . . .   | 37        |
| 5.5.8    | Festplattenplatz (disk) . . . . .                                    | 37        |
| 5.5.9    | Prozessorlast (load) . . . . .                                       | 37        |

---

|          |  |           |
|----------|--|-----------|
| 5.5.10   | Benutzerzahl (users) . . . . .                                   | 38        |
| 5.5.11   | XHost (xhost) . . . . .  | 38        |
| 5.5.12   | Prozessorinformation (cpuinfo) . . . . .                         | 38        |
| 5.5.13   | Speicherinformationen (meminfo) . . . . .                        | 38        |
| 5.5.14   | Aufstellungsort (location) . . . . .                             | 39        |
| 5.5.15   | Uptime (uptime) . . . . .  | 40        |
| 5.6      | Verwendete Perl Module . . . . .                                 | 40        |
| 5.6.1    | Net::SNMP . . . . .  | 40        |
| 5.6.2    | Tie:IxHash . . . . .   | 40        |
| 5.6.3    | Net::DNS . . . . .   | 40        |
| 5.6.4    | Net::POP3 . . . . .  | 41        |
| 5.6.5    | Net::SMTP . . . . .  | 41        |
| 5.6.6    | Net::FTP . . . . .   | 41        |
| 5.7      | Erweiterbarkeit . . . . .  | 41        |
| 5.7.1    | Anforderungen an die Erweiterbarkeit . . . . .                   | 41        |
| 5.7.2    | Umsetzung der Erweiterbarkeit . . . . .                          | 42        |
| 5.8      | Datenbank und Datenbanklayout . . . . .                          | 43        |
| 5.9      | Probleme und deren Lösung . . . . .                              | 44        |
| 5.9.1    | Ping . . . . .   | 44        |
| 5.9.2    | Bestimmung der Hardware Adresse zu einer IP<br>Adresse . . . . . | 45        |
| <b>6</b> | <b>Voraussetzungen und Installation</b>                          | <b>46</b> |
| 6.1      | Voraussetzungen . . . . .  | 46        |
| 6.2      | Installation . . . . .   | 46        |
| <b>7</b> | <b>Zusammenfassung</b>   | <b>48</b> |

|          |  |           |
|----------|--|-----------|
| <b>A</b> | <b>RFCs</b>                                      | <b>50</b> |
| A.1      | SNMPv1 Protokoll . . . . .                       | 50        |
| A.2      | SNMPv2 Protokoll . . . . .                       | 50        |
| A.3      | SNMPv3 Protokoll . . . . .                       | 51        |
| A.4      | SMIv1 Data Definition Language . . . . .         | 51        |
| A.5      | SMIv2 Data Definition Language . . . . .         | 51        |
| <b>B</b> | <b>Zuordnung von Switch Interfaces zu Räumen</b> | <b>52</b> |
| B.1      | Datenbanklayout . . . . .                        | 52        |
| <b>C</b> | <b>Beispiel eines Erweiterungsskriptes</b>       | <b>54</b> |
| <b>D</b> | <b>MIB Ressourcen</b>                            | <b>62</b> |
| <b>E</b> | <b>Erweiterung der Funktionalität</b>            | <b>63</b> |
|          | <b>Literaturverzeichnis</b>                      | <b>65</b> |
|          | <b>Danksagung</b>                                | <b>66</b> |

# Kapitel 1

## Einleitung

Der Computerarbeitsplatz ist heutzutage in vielen Organisationen und Firmen zum Standard geworden; dabei kommt es nur noch selten vor, dass die Rechner im “Standalone-Betrieb” eingesetzt werden. In den LANs (Local Area Networks) der meisten mittelständischen Unternehmen findet man inzwischen Dienste wie File- und Druckserver, Router, Firewalls und vieles mehr. Eine vernetzte Arbeitsumgebung erleichtert und optimiert auf der einen Seite die Arbeit der Angestellten und spart u.U. Kosten; auf der anderen Seite bringen immer größere Netzwerke auch immer mehr Administrationsaufwand mit sich. Je mehr Dienste oder Rechner in einem Netzwerk beheimatet sind, desto mehr können auch ausfallen. Die Aufgaben der Systemverwaltung bestehen daher zu einem Großteil darin, sicherzustellen, dass alles zur Zufriedenheit der Benutzer funktioniert und ein Ausfall unabhängig von den Gründen so schnell wie möglich behoben wird, im Idealfall sogar bevor die Benutzer überhaupt merken, dass ein Dienst ausgefallen ist.

Daher liegt es nahe, möglichst viele der essentiellen Dienste automatisch zu überwachen und der Systemverwaltung die Ergebnisse dieser Überwachung so zu präsentieren, dass ein (bevorstehender) Ausfall mit einem Blick erkennbar ist.

Ziel ist es deshalb ein umfassendes, erweiterbares Werkzeug zu konzipieren und zu implementieren, das diese Anforderungen erfüllt. Es gibt durchaus Systeme, die diese Aufgabe bereits erledigen, aber dabei handelt es sich entweder um sehr teure kommerzielle Produkte wie beispielsweise Hewlett Packards OpenView <sup>1</sup>, die selbst für mittelgroße Netze bereits zu komplex sind, oder um Lösungen, die zum Sammeln der Daten proprietäre

---

<sup>1</sup><http://www.openview.hp.com>

Protokolle nutzen, wie z.B. BigBrother <sup>2</sup> von Quest Software <sup>3</sup> oder das Open Source Projekt Spong <sup>4</sup>. Das Benutzen proprietärer Protokolle führt allerdings in vielen Netzen zu Problemen durch Interferenzen mit anderen Diensten, daher sollte gerade ein Überwachungswerkzeug darauf verzichten.

Im Rahmen dieser Arbeit wurden die Anforderungen an ein solches System anhand des Netzwerks im Mathematischen Institut der Universität Göttingen analysiert. Das Netzwerk besteht aus ca. 180 Endgeräten, die den Benutzern zahlreiche Dienste zur Verfügung stellen und ist somit als repräsentativ für ein mittelgroßes Netz anzusehen. Die Analyse, die theoretischen Überlegungen und die notwendige Recherche führten schließlich zu einer Beispielimplementation eines Überwachungs- und Informationssystems.

Aus dem oben genannten Grund basiert die Beispielimplementation komplett auf dem Simple Network Management Protocol (SNMP), das seit 1988 standardisiert ist. Die Beispielimplementation beinhaltet einige Werkzeuge, die demonstrieren sollen, welche Möglichkeiten durch SNMP geboten werden: So z.B. die Möglichkeit, vorausgesetzt die im Netz vorhandenen Swiches sind SNMP-fähig, den Ort eines Rechners zu bestimmen, was keines der professionellen Überwachungswerkzeuge bietet. Einige der Funktionalitäten basieren jedoch auf dem Einsatz bestimmter SNMP-Implementationen auf den zu überwachenden Systemen, was aber die *allgemeine* Benutzbarkeit nicht einschränkt.

---

<sup>2</sup><http://www.bb4.com>

<sup>3</sup><http://www.quest.com>

<sup>4</sup><http://spong.sourceforge.net>

# Kapitel 2

## Netzwerküberwachung und -information

### 2.1 Netzwerküberwachung

Je komplexer und inhomogener ein Netzwerk ist, desto komplizierter ist es für die Administratoren sicherzustellen, dass alle Dienste auch verfügbar sind, wenn sie gebraucht werden. Wie lässt sich die Verfügbarkeit von Diensten am besten sicherstellen ohne sie manuell in periodischen Zeitabständen selber testen zu müssen? Netzwerküberwachung ist somit in diesem Kontext gemeint als das automatisierte Prüfen von Diensten und Ressourcen und die Einleitung der entsprechenden Maßnahmen, die zur Behebung auftretender Probleme führen.

Es wird zwischen zwei Arten von Diensten unterschieden: solche, die nicht nur lokal auf einem Endgerät angeboten werden, sondern auch für andere Teilnehmer im Netz zur Verfügung stehen (z.B. Webserver, FTP-Server, SSH-Daemons, Druckserver und z.T. auch Datenbanken), und solche, die nur denjenigen Benutzern zur Verfügung stehen, die lokal mit dem entsprechenden Endgerät arbeiten, die also nicht exportiert werden. Die zweite Kategorie beinhaltet z.B. Cron oder auch einfach die Verfügbarkeit von Prozessorleistung und ist in der Regel aufwendiger zu überwachen, eben gerade weil sie nur lokal verfügbar ist.

Außerdem wird zwischen zwei Kategorien von Netzwerkendgeräten unterschieden. Zum einen sind das softwarebasierte Systeme, auf denen es möglich ist eigene der Überwachung dienende Werkzeuge zu installieren. Zum anderen sind es reine Hardwarelösungen, die einen bestimmten

Dienst anbieten. Die Überwachung solcher Endgeräte (Drucker, Videokonferenzeinheiten etc.) und deren Zustandsdaten gestaltet sich meist sehr aufwendig, da sie oft nur durch herstellerspezifische Methoden oder Webseiten erfolgen kann.

Ziel eines Netzwerküberwachungssystems ist es daher eine einheitliche Umgebung zu schaffen, mit der möglichst viele der in einem Netzwerk vorhandenen Komponenten und die Verfügbarkeit der Dienste, die sie anbieten, auf einen Blick erfasst werden können.

## **2.2 Netzwerkinformation**

Informationen über die in einem Netzwerk vorhandenen Endgeräte sind ebenso wichtig wie das Sicherstellen ihrer Funktionalität und das Überwachen der angebotenen Dienste. Die Informationen müssen dabei aufgrund ihrer Menge selektiert und kategorisiert werden, denn nicht alle Informationen sind für jeden von Interesse. Ein Netzwerkinformationssystem muss daher so konfigurierbar sein, dass verschiedene Anwender verschiedene Sichten auf die für sie interessante Informationen haben. Die folgenden zwei Abschnitte werden an Beispielen den Einsatzzweck eines Netzwerkinformationssystems erläutern.

### **2.2.1 Informationen für die Administratoren**

Die Systemverwalter müssen auf weit mehr Informationen Zugriff haben als nur die Verfügbarkeit gewisser Dienste. So wäre es z.B. wünschenswert, eine Liste der Hardwarekomponenten zu jedem Rechner zu haben, zu jeder Zeit zu wissen, in welchem Raum sich ein Rechner befindet, oder auch, wenn es sich um ein X-Terminal handelt, zu wissen, auf welcher Workstation der Benutzer des X-Terminals gerade arbeitet. Viele dieser Informationen lassen sich bei Bedarf natürlich auch auf anderem Wege beschaffen und viele werden zugegebenermaßen eher selten benötigt; vor allem, wenn die Komponenten im Netzwerk ihre Aufgaben wie gewünscht erfüllen. Jedoch sind, erfahrungsgemäß diese Informationen genau dann nicht verfügbar oder z.B. im Fall von Hardwarelisten nicht mehr aktuell, wenn man sie benötigt. Ein Netzwerkinformationssystem sorgt in einem solchen Umfeld dafür, dass die gewünschten Informationen vorhanden und aktuell sind.

### 2.2.2 Informationen für Benutzer

Ein Netzwerkinformationssystem dient aber nicht nur den Administratoren, sondern auch den Nutzern. In einem Netz beispielsweise, in dem an den meisten Arbeitsplätzen X-Terminals zum Einsatz kommen, d.h. der Rechner, der auf/unter dem Schreibtisch des Benutzers steht, stellt nur die grafische Oberfläche zur Verfügung, und der Benutzer erhält beim Start des Rechners einen sog. Hostchooser, in dem er wählen kann, auf welcher Workstation er arbeiten will (auf welchem Rechner also seine Prozesse laufen werden), ist es für den versierten Benutzern durchaus von Interesse, welche der Workstations die schnellste ist oder mit welchem Betriebssystem die einzelnen Workstations ausgestattet sind. Diese Informationen lassen sich durchaus mit Hilfe eines Netzwerkinformationssystems automatisch zur Verfügung zu stellen. Natürlich könnte man hier entgegensetzen, dass es sich dabei um Informationen handelt, die erstens nicht unbedingt essentiell wichtig sind und zweitens von einer gewissen Kontinuität sind, sodass dem Administrator durchaus zuzutrauen wäre, eine solche Informationsseite manuell zu pflegen. Aber gerade der zweite Punkt trifft nur in der Theorie zu; in der Praxis gehört eine solche Aufgabe ganz unten auf die Prioritätenliste und wird somit schnell vergessen oder vernachlässigt.

# **Kapitel 3**

## **Das Simple Network Management Protocol**

### **3.1 Was ist SNMP?**

Das Simple Network Management Protocol wurde 1988 eingeführt um die wachsende Aufgabe des Verwaltens von Internet Protokol (IP) basierten Endgeräten bewältigen zu können. SNMP bietet dem Benutzer einen einfachen Satz von Befehlen zum Verwalten dieser Geräte über ein Netzwerk.

Dieser simple Satz von Befehlen erlaubt dem Administrator relevante Daten des Endgerätes zu erfragen und bei Bedarf auch den Status eines Gerätes zu ändern.

Der Vorgänger von SNMP, das Simple Gateway Management Protocol (SGMP), wurde entwickelt, um Internet Router zu verwalten [1]; SNMP hingegen erlaubt die Überwachung und Steuerung nahezu jedes beliebigen Netzwerk-Endgerätes: von Routern über Switches und Drucker bis hin zu Workstations und Datenbanken.

### **3.2 Verschiedene SNMP Versionen und RFCs**

SNMP wird, wie andere Standardprotokolle in IP-basierten Netzen auch, von der Internet Engineering Task Force (IETF) mittels Requests for Comments (RFCs) definiert. Zur Definition von SNMP in den verschiedenen Versionen gehören eine ganze Reihe von RFCs, die im Anhang A aufgelistet

sind. Offiziell veröffentlicht werden RFCs auf den Webseiten der IETF<sup>1</sup>. Auf den Webseiten der Ohio State University <sup>2</sup> findet man einen etwas übersichtlicheren Zugang zu den RFCs.

Das SNMP wurde im Laufe der letzten Jahre weiterentwickelt und liegt inzwischen als vorgeschlagener Standard in der Version 3 vor. Der Hauptunterschied in den verschiedenen SNMP Versionen ist die Verschlüsselung der Daten auf ihrem Weg durch das Netz und die Definition der Zugriffsrechte. Version 1 (SNMPv1) ist in RFC 1157 definiert und ist von der IETF als finaler Standard eingestuft. Die Sicherheit von SNMPv1 basiert auf sog. Communities. Der Manager sendet bei jeder Anfrage einen Community-String, den der Agent mit seiner Konfiguration vergleicht. „Die Community vereint gewissermaßen die Aufgabe von Benutzername und Passwort in einem einzelnen String“ [2]. SNMPv1 bietet keinerlei Verschlüsselung, der Community-String wird also als Klartext-Zeichenkette übermittelt. Dabei wird zwischen drei verschiedenen Communities unterschieden: read-only, read-write und trap. Eine Erläuterung dessen folgt in Abschnitt 3.3.

SNMP Version 2 existiert in verschiedenen Varianten. Die am weitesten verbreitete Version heißt „community string based SNMP Version 2“ (SNMPv2c), ist in den RFCs 1905, 1906 und 1907 definiert und ist von der IETF als „experimental“ eingestuft. Trotz dieser Einstufung wird es von vielen Herstellern in der Praxis bereits unterstützt. SNMPv2c bietet unter anderem einen etwas erweiterten Befehlssatz (siehe 3.3) nutzt aber den selben Authentifizierungsmechanismus wie SNMPv1, daher auch der Name „community based“. Die sicherheitstechnischen Aspekte betrachtet bietet SNMPv2 also keinerlei Vorteile.

SNMP Version 3 (SNMPv3) ist die nächste Version des Protokolls, die Standard Status erreichen wird. Definiert in den RFCs 1905, 1906, 1907, 2571, 2572, 2573, 2574 und 2575, fügt es neue Authentifizierungsmechanismen und komplette Verschlüsselung der übertragenen Daten hinzu. Die Verbreitung von SNMPv3 ist jedoch eher als sehr gering zu bewerten. Lediglich einige Software SNMP-Agenten zum Einsatz auf Workstations unterstützen bereits diese Version.

---

<sup>1</sup><http://www.ietf.org/rfc>

<sup>2</sup><http://www.cis.ohio-state.edu/cs/Services/rfc>

### 3.3 Agenten, Manager und deren Kommunikation

Im Zusammenhang mit SNMP spricht man nicht wie in anderen Bereichen von Clients und Servern, doch es handelt sich um ein ähnliches Modell.

Wie im Client-Server-Modell hat man es mit zwei Arten von Instanzen zu tun: Agenten und Manager. Ein Manager, auch oft mit Management Host oder Network Management Station (NMS) bezeichnet, ist ein Server mit einer Software, die in der Lage ist, andere Geräte mit Hilfe des SNMP zu verwalten (manage). D.h., der Manager kann Daten von den zu verwaltenden Endgeräten abrufen oder auf deren Nachrichten reagieren. Die zweite Art von Instanzen wird Agent genannt. Ein Agent ist ein Programm auf dem zu überwachenden Gerät, welches die Daten sammelt, die es zur Beantwortung der Abfragen des Managers benötigt, oder die Requests des Managers zur Zustandsänderung in die Tat umsetzt. Ein Agent kann auch von sich aus auf vordefinierte Ereignisse hin den Manager informieren.

Manager und Agenten kommunizieren, wie bereits einleitend erwähnt, durch einen einfachen Satz von Befehlen miteinander. In SNMPv1 sind nur fünf Befehle definiert durch die die Kommunikation erfolgt. Drei dieser Befehle kann der Manager an den Agenten senden:

- **GET-Request:** Anfrage an den Agenten nach einem Variablenwert
- **GET-NEXT-Request:** Anfrage an den Agenten nach dem Wert der nächsten Variablen, ohne deren Namen kennen zu müssen
- **SET-Request:** Anweisung an den Agenten, einen Variablenwert zu ändern

Die anderen beiden Befehle gehen von Agenten aus:

- **GET-Response:** als Antwort aus die oben genannten Requests
- **TRAP:** bietet die Möglichkeit, dass der Agent von sich aus den Manager über ein Ereignis informiert

SNMPv2 und SNMPv3 erweitern diesen Befehlssatz um zwei Befehle und benennen zwei um:

- **GET-BULK-Request:** gleichzeitige Abfrage mehrerer Variablenwerte
- **NOTIFICATION:** die SNMPv2 Version eines TRAP
- **INFORM:** wird von Agenten ausgelöst wie TRAP, erwartet aber eine Antwort vom Manager
- **REPORT:** in SNMPv3 eingeführt, dient der Befehl zur Kommunikation zwischen Managern

## **3.4 Position von SNMP in der TCP/IP-Suite und die daraus resultierenden Folgen**

SNMP Befehle nutzen das User Datagram Protocol (UDP) zur Übermittlung der Daten. Im OSI-Schichtenmodell sind SNMP Manager und Agenten also der Applikationsschicht zuzuordnen und somit unabhängig von der zugrundeliegenden Hardware [3]. UDP wurde ursprünglich gewählt, weil es ein verbindungsloses Protokoll ist und der Overhead, der durch die Verwendung eines verbindungsorientierten Protokolls (des Transmission Control Protocols - TCP) entstanden wäre [4], für die damaligen Netze durchaus eine Belastung gewesen wäre (20 statt 8 Byte pro gesendetem Datenpaket).

Die Benutzung des UDP zur Übertragung der Daten bringt allerdings Folgen mit sich: Es kann nicht sichergestellt werden, dass abgesetzte Befehle ihr Ziel erreichen. Dadurch könnte man sogar behaupten, dass es sich bei SNMP um ein unzuverlässiges Protokoll handelt.

Auf der einen Seite spielt diese Unzuverlässigkeit keine Rolle: Wenn der Manager einen GET-Request absetzt und keine Antwort erhält, weil entweder der Request selbst oder aber die Antwort verloren geht, so lässt sich das Problem durch einen simplen Timeout lösen, nach dessen Ablauf der Manager den Befehl einfach wiederholt. Das ist in nahezu allen UDP basierten Anwendungen die gängige Praxis.

Auf der anderen Seite ergibt sich aber ein Problem, wenn ein Agent einen TRAP absetzt. Auf diesen erwartet er keine Antwort; dennoch kann er nicht sicher gehen, dass die (vielleicht wichtigen) Informationen tatsächlich angekommen sind. SNMPv2 löst dieses Problem durch die Einführung des INFORM Befehls.

Ein weiterer Grund, der für die Verwendung von UDP statt TCP spricht, ist, dass es sich bei SNMP um ein Werkzeug zur Überwachung von Netzwerkkomponenten handelt. Mauro und Schmidt beschreiben die Situation in O'Reillys Standardwerk zu SNMP [1] wie folgt: „Befände man sich in einem unfehlbaren Netz, so wäre ein solches Werkzeug gar nicht erst nötig, da Netzwerke aber nun mal von Natur aus nicht unfehlbar sind, muss man sie in Problemsituationen nicht noch dadurch weiter belasten, dass das Überwachungswerkzeug unaufhörlich versucht Netzwerkverbindungen aufrecht zu erhalten.“

## 3.5 Verwaltete Informationen

SNMP ist mehr als die Definition von Kommunikationsprotokollen. Weiterer zentraler Bestandteil ist die Beschreibung der verwalteten Daten. RFC 1155 (The Structure of Management Information Version 1 - SMIV1) definiert die Struktur, die Benennung und die Datentypen der verwalteten Objekte.

SMI Objekte lassen sich mit Hilfe dreier Attribute beschreiben:

**Name:** Der Name, auch Object Identifier (OID) genannt, definiert das zu verwaltende Objekt eindeutig. OIDs tauchen in zwei Formen auf: numerisch und 'menschlesbar'. In beiden Fällen sind sie aber lang und umständlich zu handhaben.

**Syntax und Typen:** Der Datentyp eines verwalteten Objekts wird durch eine Untermenge der Abstract Syntax Notation One (ASN.1) definiert. ASN.1 ist plattformunabhängig und spezifiziert, wie Daten dargestellt und über Netzwerke übertragen werden.

**Kodierung:** Jede Instanz eines Objekts wird durch Benutzung der Basic Encoding Rules (BER) in eine Folge von Octets kodiert. Die BER geben dabei an, wie Objekte zum Transport über ein Medium wie z.B. Ethernet kodiert und dekodiert werden.

### 3.5.1 Der SNMP Namensraum

Die verwalteten Objekte sind in einer baumähnlichen Struktur organisiert, die die Basis für die Namesgebung bildet. Ein Objektname besteht aus einer Reihe durch Punkte getrennter, ganzzahliger, positiver Zahlen, basierend auf den Knoten des Baumes. Die menschenlesbare Variante der OIDs ist ebenfalls nicht mehr als eine Reihe von Namen, getrennt durch Punkte, die man als die Namen der Knoten ansehen kann. Die Blätter dieses Baumes repräsentieren die eigentlich zu verwaltenden Objekte.

Der komplette Baum besteht aus weit mehr als den für den Umgang mit SNMP relevanten Knoten. Die Wurzel des Baums hat keinen Namen und enthält z.Zt. drei Unterbäume: *ccitt(0)* - verwaltet von dem International Telegraph and Telephone Consultative Committee, *iso(1)* - verwaltet von der International Organisation for Standardization und *joint(2)* - verwaltet von beiden Organisationen gemeinsam.

Der im Zusammenhang mit SNMP interessante Ast ist der *iso*-Unterbaum, der diverse Unterbäume hat. Einer davon ist *org(3)*.

Unter *iso(1).org(3)* findet sich unter anderem ein Knoten mit der Bezeichnung *dod(6)* (*dod* steht für das U.S. Departement of Defense) der wiederum diverse Unterbäume hat. Der hier interessante ist der *internet(1)* Unterbaum. Kurz gesagt finden sich alle für SNMP relevanten Objekte unterhalb von *iso.org.dod.internet (1.3.6.1)*.

Um sicherzustellen, dass alle Objekte im OID Baum eindeutig sind, sind verschiedene Organisationen für die einzelnen Unterbäume zuständig; im Fall des *internet*-Unterbaumes ist es das Internet Activities Board (siehe RFC 1160).

Der *iso.org.dod.internet* Ast ist unterteilt in vier Unterbäume: *directory(1)*, *mgmt(2)* kurz für *management*, *experimental(3)* und *private(4)*. Der *directory*-Ast wird z.Zt. nicht benutzt, der *mgmt*-Unterbaum beinhaltet die standardisierten Objekte zur Verwaltung durch SNMP. Der *experimental*-Ast ist zu Forschungszwecken reserviert und der *private*-Ast enthält z.Zt. nur einen weiteren Unterbaum: *enterprises(1)*, in welchem Unternehmen sowie Privatpersonen von der Internet Assigned Numbers Authority (IANA) <sup>3</sup> eigene Äste zugewiesen bekommen können. Dort finden sich viele herstellereigenspezifische Managementobjekte.

### 3.5.2 Definition von Objekten

Bevor die Objekte selbst definiert werden können, müssen zunächst die Datentypen definiert sein; SMIV1 definiert zwölf Datentypen, die in Tabelle 3.1 auf Seite 16 beschrieben sind.

Die Definition der Baumstruktur und der Blätter erfolgt wie bereits erwähnt durch SMIV1. Als Beispiel sei hier zunächst die Definition des *internet* Unterbaums und seiner vier Unterbäume angegeben:

```
internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

Die Definitionen weiterer Unterbäume und vor allem der darin enthaltenen Blätter, die im Endeffekt die Objekte definieren, um deren Verwaltung es geht, finden sich in sog. Management Information Bases (MIBs).

---

<sup>3</sup><http://www.iana.org>

| Datentyp          | Beschreibung   |
|-------------------|--|
| OCTET STRING      | Ein String mit null oder mehr Octets - wird meist für Text genutzt.  |
| INTEGER           | Eine 32-Bit-Zahl, die oft auch zur Repräsentation von Statustypen genutzt wird. (Anmerkung: Nach RFC 1155 darf die Null nicht zur Beschreibung eines Status genutzt werden.)   |
| COUNTER           | Eine 32-Bit-Zahl zwischen 0 und $2^{32} - 1$ einschließlich. Eine COUNTER-Variable kann nur erhöht werden; bei Erreichen des Maximums beginnt sie automatisch wieder bei Null. |
| OBJECT IDENTIFIER | Ein durch Punkte getrennter String aus ganzen Zahlen, der einen Verweis auf ein Objekt repräsentiert.  |
| SEQUENCE          | Eine Liste, die null oder mehr ASN.1 Datentypen enthält.   |
| SEQUENCE OF       | Ein Objekt, welches aus einer SEQUENCE von ASN.1 Datentypen besteht.   |
| IpAddress         | Eine 32-Bit IPv4 Adresse.  |
| NetworkAddress    | Wie IpAdress, kann aber mehrere Netzwerkadrestypen enthalten.  |
| Gauge             | Wie COUNTER ebenfalls 32 Bit. GAUGE kann aber erhöht <i>und</i> verringert werden, kann jedoch Minimum und Maximum nicht überschreiten.  |
| TimeTicks         | Eine 32-Bit-Zahl zwischen 0 und $2^{32} - 1$ einschließlich, die zur Speicherung von Zeiten genutzt wird. TimeTicks speichert Zeiten in hundertstel Sekunden.                  |
| Opaque            | Erlaubt das Einpacken jedes beliebigen ASN.1 Typs in einen OCTET STRING.   |
| NULL              | Wird zur Zeit von SNMP nicht genutzt.  |

Tabelle 3.1: Datentypen in SMIV1

„Die grundlegende SNMP MIB (MIB-I) wurde in RFC1066 spezifiziert und beschreibt den Zugriff auf grundlegende Verwaltungsdaten: Informationen über das System, seine Schnittstellen, Adressumformung und Protokollinformationen. Später wurde eine vollständigere Überarbeitung der MIB bereitgestellt, MIB-II, die in RFC1213 definiert ist“ [5]. Die MIB-II zählt zu den wichtigsten MIBs, da sie die einzige ist, die auf allen Endgeräten implementiert sein muss, damit diese sich SNMP-fähig nennen dürfen [6] .

Die Definition weiterer Äste und der Blätter innerhalb einer MIB erfolgt ebenfalls durch ASN.1.

Äste werden dabei definiert wie oben bereits gezeigt. Die Definition von Objekten sieht folgendermaßen aus:

```
<Name> OBJECT-TYPE
  SYNTAX <Datentyp>
  ACCESS <entweder read-only, read-write,
        write-only oder not-accessible>
  STATUS <entweder mandatory, optional oder obsolete>
  DESCRIPTION
    "Textuale Beschreibung des Objekts"
  ::= { <Eindeutige OID dieses Objekts> }
```

### Beispieldefinitionen

Da das Verstehen und Lesen können von MIBs eine der Grundvoraussetzungen zum Umgang mit SNMP ist, soll dies noch einmal an zwei Beispielen aus der MIB-II erläutert werden. Jede MIB erhält zunächst einen Namen:

```
RFC1213-MIB DEFINITIONS ::= BEGIN
```

In diesem Fall lautet der Name RFC1213-MIB. Es folgen Import-Anweisungen zur Übernahme von Teilen aus anderen MIBs:

```
IMPORTS
    mgmt, NetworkAddress, IpAddress,
    Counter, Gauge, TimeTicks
    FROM RFC1155-SMI
OBJECT-TYPE
    FROM RFC-1212;
```

und die Definitionen der verwendeten Äste:

```
mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }

-- groups in MIB-II

system     OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces OBJECT IDENTIFIER ::= { mib-2 2 }
at         OBJECT IDENTIFIER ::= { mib-2 3 }
```

```

ip          OBJECT IDENTIFIER ::= { mib-2 4 }
icmp       OBJECT IDENTIFIER ::= { mib-2 5 }
tcp        OBJECT IDENTIFIER ::= { mib-2 6 }
udp        OBJECT IDENTIFIER ::= { mib-2 7 }
egp        OBJECT IDENTIFIER ::= { mib-2 8 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp       OBJECT IDENTIFIER ::= { mib-2 11 }

```

Danach erfolgen die eigentlichen Objektdefinitionen, z.B.:

```

sysDescr OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE (0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "A textual description of the entity.
        This value should include the full
        name and version identification of
        the system's hardware type, software
        operating-system, and networking
        software. It is mandatory that this
        only contain printable ASCII
        characters."
    ::= { system 1 }

```

Die Systembeschreibung des zu überwachenden Geräts würde man also unter *iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0* bzw. *1.3.6.1.2-1.1.1.0* finden. Hier sei auf die „0“ am Ende hingewiesen: In SNMP sind MIB Objekte immer in der Form *x.y* definiert, wobei *x* die eigentliche OID des Objekts und *y* der Index der Instanz des Objekts ist. Im folgenden wird deutlich, warum zwischen verschiedenen Instanzen unterschieden werden muss.

Komplizierter werden die Definitionen, wenn es um die Beschreibung von Sequenzen oder Tabellen geht:

```

ifTable OBJECT-TYPE
    SYNTAX SEQUENCE OF IfEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION

```

```

        "A list of interface entries. The
        number of entries is given by the
        value of ifNumber."
 ::= { interfaces 2 }

ifEntry OBJECT-TYPE
    SYNTAX IfEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "An interface entry containing
        objects at the subnetwork layer
        and below for a particular
        interface."
    INDEX { ifIndex }
    ::= { ifTable 1 }

IfEntry ::=
    SEQUENCE {
        ifIndex
            INTEGER,
        ifDescr
            DisplayString,
        ifType
            INTEGER,
        .
        .
        .

        ifSpecific
            OBJECT IDENTIFIER
    }

ifIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "A unique value for each interface.
        Its value ranges between 1 and the
        value of ifNumber. The value for
        each interface must remain constant
```

```

        at least from one re-initialization
        of the entity's network management
        system to the next reinitialization."
 ::= { ifEntry 1 }

```

*ifTable* beschreibt eine Tabelle aller Netzwerk-Interfaces des zu verwalten- den Gerätes, bei Switches z.B. gibt es pro Port einen Eintrag in dieser Ta- belle, bei Unix/Linux Workstations einen Eintrag für jedes Device, egal ob Modem, Netzwerkkarte oder Loopback. *ifTable* ist definiert als SEQUENCE OF *IfEntry*, also als Liste; jeder Eintrag in dieser Liste ist eine Zeile in der Tabelle. *IfEntry* wiederum ist definiert als SEQUENCE von unterschiedli- chen Datentypen, die den Spalten in der Tabelle entsprechen. Jeder Spalte ist ein Name und somit eine Objektdefinition zugeordnet. Die Beschreibung (*ifDescr*) des zweiten Interfaces, also der zweiten Instanz von *IfEntry*, würde man also unter *iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.if- Descr.2* bzw. *1.3.6.1.2.1.2.2.1.2.2* finden.

### 3.5.3 Erweiterungen durch SMIV2

SMIV2 erweitert die SMI Definitionen durch Hinzufügen eines Astes na- mes *snmpV2* zum *internet*-Unterbaum, der diverse neue Datentypen und andere Änderungen enthält und ist in den RFCs 2578, 2579, 2580 defi- niert. Die komplette OID für diesen Ast lautet *iso.org.dod.internet.snmpv2.- snmpModules.snmpMIB.snmpMIBObjects* oder *1.3.6.1.6.3.1.1*. Tabelle 3.2 enthält die in SMIV2 neuen Datentypen.

| Datentyp   | Beschreibung   |
|------------|--|
| Integer32  | Dasselbe wie INTEGER.  |
| Counter32  | Dasselbe wie COUNTER.  |
| Gauge32    | Dasselbe wie Gauge.  |
| Unsigned32 | Dezimalwerte zwischen 0 und $2^{32} - 1$ einschließlich.     |
| Counter64  | Wie Counter32, aber mit einem Maximalwert von $2^{64} - 1$ . |
| BITS       | Eine Aufzählung von benannten Bits.                          |

Tabelle 3.2: Datentypen in SMIV2

Auch die Definition von Objekten hat sich in SMIV2 leicht geändert. Es wur- den vier zusätzliche optionale Felder für Objektdefinitionen eingeführt, die bessere Beschreibungen, zusätzliche Tabellenspalten und bessere Zugriffs- kontrolle erlauben [7]:

- **MAX-ACCESS:** Alternative zum ACCESS Schlüsselwort. Gültige Optionen sind: read-only, read-write, read-create, not-accessible und accessible-for-notify.
- **STATUS:** Die gültigen Optionen sind erweitert worden und schließen jetzt CURRENT, OBSOLETE und DEPRECATED ein, wobei CURRENT in SNMPv2 die Bedeutung von MANDATORY übernimmt.
- **UnitsParts:** Textuale Beschreibung der Einheit, die das Objekt definiert.
- **AUGMENTS:** Ermöglichung der Erweiterung einer Tabelle durch Hinzufügen zusätzlicher Spalten.

Zusätzlich definiert SMIV2 15 Textkonventionen, die eine abstraktere Definition von Objekten erlauben; Tabelle 3.3 beschreibt diese Definitionen.

| Name            | Beschreibung  |
|-----------------|---|
| DisplayString   | Ein aus ASCII-Zeichen bestehender String von nicht mehr als 255 Zeichen Länge.  |
| PhysAddress     | Eine durch einen OCTET STRING dargestellte physikalische Adresse.   |
| MacAddress      | Sechs Octets zur Beschreibung einer MAC nach IEEE 802 (Hardware Adresse einer Ethernet Karte).  |
| TruthValue      | Definition von Boolean Werten.  |
| TestAndIncr     | Verhindert, dass zwei oder mehr Management Hosts ein Objekt gleichzeitig bearbeiten.  |
| AutonomousType  | Ein Object Identifier zur Definition eines Unterbaumes mit zusätzlichen Definitionen.   |
| VariablePointer | Ein Zeiger auf eine bestimmte Instanz eines Objekts.  |
| RowPointer      | Ein Zeiger auf eine Tabellenzeile.  |
| RowStatus       | Überwacht den Status und die Konsistenz einer Tabelle, insbesondere, wenn mehrere Manager darauf zugreifen und die Spalten einer Tabelle verändern. |
| TimeStamp       | Misst die Zeit zwischen der "system uptime" und einem bestimmten Ereignis.  |
| TimeInterval    | Beschreibt eine Zeitperiode.  |
| DateAndTime     | Beschreibt Datum und Zeit durch einen OCTET STRING.   |
| StorageType     | Beschreibt die Art von Speicher, die ein Agent benutzt; mögliche Werte: other(1), volatile(2), nonVolatile(3), permanent(4) und readOnly(5).        |
| TDomain         | Beschreibt eine Art von Transport Service.  |
| TAddress        | Beschreibt eine Adresse des Transport Service.  |

Tabelle 3.3: Textkonventionen in SMIV2

# **Kapitel 4**

## **Überblick über Implementation und Komponenten**

### **4.1 Überblick über die Implementation**

Die Software besteht aus zwei im Prinzip völlig unabhängigen Teilen. Der erste Teil liest die Konfigurationsdateien, sammelt die gewünschten Informationen, stellt die Präsentation der Informationen zusammen und trägt Änderungen der Zustände in die Datenbank ein. Die letzten drei Schritte werden nach einem vorgegebenen Intervall wiederholt. Der zweite Teil dient der Präsentation vergangener Zustände. Diese Informationen werden bei Bedarf aus der Datenbank geholt und für den Benutzer aufbereitet.

### **4.2 Die Komponenten**

Im Folgenden werden die Komponenten der Implementation und die Kriterien für deren Auswahl erläutert. Hieraus ergeben sich auch die Voraussetzungen an das System, auf dem die Überwachungssoftware eingesetzt werden soll.

#### **4.2.1 Implementation in Perl**

Als Implementationssprache wurde aus diversen Gründen Perl gewählt. Zum einen ist Perl Bestandteil jeder Linux Distribution und auch für alle anderen Betriebssysteme ist es kostenfrei erhältlich. Zum anderen ist Perl im

Vergleich zu manchen anderen Sprachen, in denen sich objekt-orientierte Ansätze verwirklichen lassen, recht schnell. Auch wenn objekt-orientierte Programmierung in Perl nicht so sauber ist wie das bei anderen Sprachen der Fall wäre, so bietet Perl dem Entwickler trotzdem genug Möglichkeiten für ein solches Projekt. Ein weiterer Vorteil von Perl ist die Verfügbarkeit von Erweiterungen; es gibt eine kostenfreie, gute und vollständige SNMP Bibliothek, was bei vielen anderen Sprachen nicht der Fall ist, und für Datenbankverbindungen ist Perl geradezu prädestiniert.

### 4.2.2 NetSNMP

NetSNMP ist eine SNMP Implementation für Linux/Unix Systeme, die sowohl einen Agenten als auch einfache Werkzeuge für einen Management Host mitliefert. Nachdem das Projekt einer SNMP Entwicklung für diese Plattformen an der Carnegie Mellon University (CMU) nicht mehr weiterentwickelt wird, ist NetSNMP die z.Zt. vollständigste freie Implementation für Linux/Unix. NetSNMP ist aus einem Projekt an der University of California in Davis hervorgegangen und hieß bis vor Kurzem auch noch ucdSMNP. Hinweise darauf findet man noch in vielen Teilen des Projekts: beispielsweise sucht die Agenten-Software in vielen Distributionen noch nach einer Konfigurationsdatei mit dem Namen `ucdsnmpd.conf` und der Unterbaum für die Enterprise Erweiterungen findet sich unter `iso.org.dod.internet.private.enterprises.ucdavis` NetSNMP wird in diesen Projekt nur auf der Seite des Agenten eingesetzt, da managerseitig die Perl-SNMP-Bibliothek zum Einsatz kommt.

### 4.2.3 MySQL

Viele kommerzielle Management Software Produkte benutzen - wenn überhaupt - proprietäre Datenbanken. Für dieses Projekt war eine Datenbank notwendig. In der Entwicklungsumgebung war eine MySQL Datenbank vorhanden, und Perl bietet uneingeschränkten Zugriff auf eine solche. Im Prinzip wäre aber jede andere SQL-Datenbank genauso gut gewesen.

### 4.2.4 PHP

PHP wird verwendet, um die Daten vergangener Zustandsänderungen aus der Datenbank zu holen und für den Betrachter aufzubereiten. PHP wurde

deshalb gewählt, weil es im Gegensatz zu Perl nicht das Common Gatewas Interface (CGI) nutzt, sondern von vielen Webservern direkt interpretiert wird. In vielen Umgebungen wird bewußt auf den Einsatz von CGI und damit auch auf die Möglichkeit einen Webserver ein Perl-Skript ausführen zu lassen verzichtet. Dies geschieht zumeist aus sicherheitstechnischen Aspekten.

PHP hingegen wird auf den meisten Systemen geduldet, wenn nicht sogar bereits genutzt. Zudem müsste das Überwachungsprogramm zur Benutzung von Perl zum Auslesen der Daten aus der Datenbank Schreibrechte im cgi-bin Verzeichnis haben, was von vornherein für viele Administratoren keine Option wäre. PHP-Skripte hingegen müssen nicht in ein gesondertes Verzeichnis, sie werden mit den Standard HTML-Dateien zusammen in einem Verzeichnis abgelegt und von dort aus ausgeführt. Die Datenbankfunktionalitäten von PHP stehen denen von Perl in nichts nach, sodass sich PHP für den hier vorliegenden Einsatzzweck ebenso eignet.

#### **4.2.5 Präsentation der gesammelten Daten**

Die Präsentation der gesammelten Informationen, sowohl für den Administrator als auch für den Anwender, erfolgt über HTML-Seiten, die zum größten Teil direkt bei Erhalt der Daten von den Perl-Skripten generiert werden. Einige Teile, wie z.B. die vergangenen Zustände eines Dienstes auf einem überwachten Endgerät, werden erst auf Anfrage aus den entsprechenden Datenbank-Einträgen generiert, da die Anzahl der HTML-Dateien und somit der verbrauchte Speicher sonst sehr schnell sehr groß werden würde.

Als Präsentationsmedium wurden HTML-Seiten gewählt, da die Voraussetzungen für die Nutzung der Software damit gering gehalten werden. Webbrowser sind auf jedem System vorhanden, wohingegen eine andere Präsentationsform immer Einschränkungen mit sich gebracht hätte. Einzige weitere Voraussetzung zur Präsentation der Daten ist daher ein Webserver mit Zugriff auf die Verzeichnisse, in denen der Management Host seine Daten ablegt.

# Kapitel 5

## Die Implementation im Detail

### 5.1 Testmethoden

#### 5.1.1 „Reale Tests“ vs. SNMP-Queries

Trotz der vielen Vorteile, die SNMP bietet, gibt es Situationen, in denen andere Wege zur Beschaffung von Informationen über die Verfügbarkeit von Diensten zu bevorzugen sind. Dies ist genau dann der Fall, wenn es sich bei dem zu überwachenden Dienst um einen Netzwerkdienst handelt; wenn also von anderen Punkten im Netzwerk auf ihn zugegriffen wird. Als Beispiel sei hier ein Webserver genannt: Ein SNMP-Agent auf der Workstation, auf der auch der Webserver läuft, ist durchaus in der Lage einem Manager mitzuteilen, ob der entsprechende Prozess läuft oder nicht. Im Zweifelsfall existiert dieser Prozess aber noch, arbeitet aber nicht mehr innerhalb der vorgegebenen Parameter. An dieser Stelle ist zu überlegen, ob die Überwachungssoftware anstatt den SNMP-Agenten zu fragen, nicht besser direkt eine Anfrage an den Webserver stellt, und prüft, ob die Antwort dem entspricht, was erwartet wird. Die so gewonnene Information ist in jedem Fall wertvoller. Also ist diese Methode in solchen Situationen dem Umweg über den SNMP-Agenten vorzuziehen. Ähnliches ergibt sich für Dienste wie FTP, DNS, POP3 etc.

#### 5.1.2 SNMP basierte Tests

Die über SNMP verfügbaren Informationen hängen von Agenten des zu überwachenden Systems ab, daher sind nicht alle Tests für jedes Endgerät sinnvoll. Die Object ID *1.3.6.1.2.1.25.1.5 (host.hrSystem.-hrSystemNumUsers)* z.B. liefert die Anzahl der gerade auf einem System

eingeloggten Benutzer - eine Information, die bei Workstations durchaus von Interesse ist, bei Endgeräten wie Druckern aber belanglos und deshalb oft auch gar nicht in der MIB des Agenten vorhanden.

Bei jedem zu überwachenden Endgerät wird geprüft, ob ein SNMP-Agent auf Anfragen antwortet. Das Ergebnis dieses Tests wird dem Betrachter der Statusseiten aber nur dann präsentiert, wenn in der Konfigurationsdatei danach verlangt wurde. Das Ergebnis ist aber für das weitere Testen eines Geräts von Bedeutung, da die SNMP basierten Tests an diesem Gerät gar nicht erst durchgeführt, sondern direkt mit einem entsprechenden Status versehen werden.

## 5.2 Skripte zur Ergänzung von NetSNMP

Wie bereits erwähnt, liefert NetSNMP die z.Zt. umfangreichste Implementation eines SNMP-Agenten für Linux Workstations. Aber auch ein NetSNMP-Agent liefert nicht alle Informationen, die evtl. von Interesse sind. Zum einen ist eine bestimmte MIB vielleicht nicht Bestandteil des Agenten und zum anderen sind manche OID-Werte auf Standardwerte gesetzt, da die Information bei unterschiedlichen Unix Derivaten an unterschiedlichen Stellen zu finden ist und deshalb bei der Implementation ganz darauf verzichtet wurde. Beispielsweise ist OID *host.hrDevice.hrDeviceTable* eine Tabelle mit Einträgen für die Hardwarekomponenten eines Rechners. Zum Prozessor liefert ein NetSNMP Agent aber nur "An electronic chip that makes the computer work", anstatt, wie vielleicht erwünscht, Prozessortyp und Taktfrequenz.

Genau für diese Fälle bietet NetSNMP die Möglichkeit zur Ausführung externer Programme, die dann die entsprechenden Daten sammeln und an den Agenten zurückgeben, der sie dann wiederum an den anfragenden Manager weiterleitet. Die Ergebnisse dieser Abfragen werden dem Manager innerhalb der UCD Enterprise-MIB *1.3.6.1.4.1.2021 (enterprises.ucdavis)* zur Verfügung gestellt. Der folgende Abschnitt beschreibt neben der grundsätzlichen Konfiguration von NetSNMP auch die Einbindung solcher Programme.

## 5.3 Konfiguration von NetSNMP

NetSNMP erlaubt eine sehr umfangreiche Konfiguration des Agenten (snmpd genannt). Die Konfigurationsdatei findet sich unter */etc* und heißt aus historischen Gründen, wenn nicht anders angegeben, *ucdsonmpd.conf*.

Die ersten vier Abschnitte [A-D] dienen der Zugriffskontrolle. Abschnitt A legt die Community Strings und sog. Security Names für Zugriffe aus verschiedenen Netzbereichen fest. In der Regel wird es wohl reichen den Zugriff auf das lokale Subnetz zu beschränken. Aber auch eine Beschränkung auf nur eine Adresse (die des Management Hosts) ist möglich.

```
#####
# Section A: Access Control, define security names
#####
#      name      source      community
com2sec local      localhost    public
com2sec manager    192.168.0.5 private
com2sec mysubnet   192.168.0.1/32 public
com2sec world      default     public
```

Der folgende Abschnitt fasst die Security Names und die Zugriffsmethode (SNMP Version) zu Gruppen zusammen:

```
#####
# Section B: Map security names to groups
#####
#      name      model  name
group untrusted v1     world
group untrusted v2c    world
group R0group   v1     mysubnet
group R0group   v2c    mysubnet
group RWgroup   v1     manager
group RWgroup   v2c    manager
group RWgroup   v1     local
group RWgroup   v2c    local
```

Abschnitt C erzeugt verschiedene Ansichten des kompletten Baumes, d.h. nur bestimmte Teile des Baumes werden sichtbar gemacht:

```
#####
# Section C: create a view to let the groups
#      have rights to ...
#####
#      name      type      subtree
view all      included  .1
view descr    included  system.sysDescr
```

Um die Konfiguration der Zugriffskontrolle abzuschließen, wird in Abschnitt D festgelegt, welche Gruppen mit welchen Rechten auf welche Views zugreifen dürfen:

```
#####
# Section D: grant the groups access to the views
#####
#      name      context model level  match read  write notif
access untrusted ""      any   noauth exact descr none  none
access R0group  ""      any   noauth exact all  none  none
access RWgroup  ""      any   noauth exact all  all   none
```

Abschnitt E definiert die Werte, die in der *system* MIB unter *syslocation* und *syscontact* abrufbar sind:

```
#####
# Section E: System information
#####
syslocation My Organisation
syscontact root@mydomain.com
```

NetSNMP kann die Existenz bestimmter Prozesse und deren Instanzen überwachen. Unter Angabe einer minimalen und einer maximal erlaubten Anzahl von Instanzen wird definiert, wann der Agent den Manager durch einen SNMP-Trap informieren soll. Die Informationen können aber nicht nur durch einen Trap zum Manager gelangen; der NetSNMP-Agent stellt sie auch unter *enterprises.ucdavis.prTable* zur Verfügung.

```
#####
# Section F: process checks
#####
#   procname  max  min
proc sshd    5    1
proc lpd     5    1
```

Wie bereits erwähnt, bieten NetSNMP-Agenten die Möglichkeit zur Ausführung externer Programme. Das Schlüsselwort “exec” definiert einen solchen Programmaufruf. Der Rückgabewert und die erste Zeile der Standardausgabe der so aufgerufenen Programme ist unter *enterprises.ucdavis.extTable* zu finden.

Diese Methode der Informationsbeschaffung ist allerdings dadurch begrenzt, dass eben nur die erste Zeile der Ausgabe aufgerufener Programme zurückgegeben wird; alle weiteren Zeilen werden ignoriert und sind nicht vom Manager abfragbar.

```
#####
# Section G: executables / scripts
#####
#   name      script  arguments
exec anyName  progName progArguments
```

Eine Erweiterung dessen bietet das “exec” Schlüsselwort, wenn eine MIB-Nummer mit angegeben wird. In dem Fall findet man unter *mibnum.100* den Rückgabewert des Programms und unter *mibnum.101* eine Tabelle mit einer Zeile der Programmausgabe pro Tabellenzeile.

```
#####
# Section H: extensibles
#####
#   mibnum          name      script          arguments
exec .1.3.6.1.4.1.2021.51  cpuinfo  /tmp/cpuInfo
exec .1.3.6.1.4.1.2021.52  cpumhz   /tmp/cpuMHz
```

Der NetSNMP-Agent bietet zudem Objekte zur bequemen Überwachung des Festplattenplatzes an. Abhängig vom angegeben Pfad wird ein Eintrag in die unter der OID *enterprises.ucdavis.dskTable* zu findende Tabelle gemacht, in der unter anderem der freie, der belegte und der gesamte Plattenplatz vermerkt sind. Bei Angabe von Minimalwerten (absolut oder prozentual) für den freien Platz wird der NetSNMP-Agent bei Unterschreiten der gewünschten Grenze in der Tabelle entsprechende Felder setzen.

```
#####
# Section I: disk checks
#####
#   path      minpath|minpercent%
disk /
disk /var    20000
disk /home   10%
```

Der letzte Konfigurationsparameter bietet die Möglichkeit, Grenzwerte für die “Load Average” einer Workstation zu setzen. Das Abfragen der Load Averages erlaubt der NetSNMP-Agent auch ohne spezielle Konfiguration unter der OID *enterprises.ucdavis.laTable*. Jedoch werden dann keine Fehlermeldungen erzeugt. Bei Angabe von Maximalwerten mit dem Schlüsselwort “load” erzeugt der NetSNMP-Agent Fehlermeldungen in den entsprechenden Spalten der *laTable*.

```
#####
# Section I: load average checks
#####
#   1min  5min  15min
load 5    4    4
```

## 5.4 Implementation des Perl-Teils

Die Implementation des ersten Teils der Software, der die Informationen sammelt, beruht auf einem objekt-orientierten Ansatz in Perl.

### 5.4.1 Die Konfigurationsdateien

Eine zentrale Konfigurationsdatei definiert die zur Überwachung notwendigen Parameter. Die Datei wird, wenn nicht über den Aufrufparameter *-c* angegeben, im Verzeichnis *etc* unterhalb des Verzeichnisses des Programms und in */etc* gesucht.

Die Datei muss folgende Definitionen enthalten:

```
webdir=/xx/xx
```

Angabe des Verzeichnisses, in dem die HTML-Dateien zur Präsentation der Ergebnisse abgelegt werden sollen.

```
dbdriver=DRIVERNAME
dbhost=HOSTNAME
dbuser=USERNAME
dbpasswd=PASSWORD
```

Angabe des Datenbanktreibers und der Zugangsdaten zur Datenbank, in der die Ergebnisse gesichert werden sollen.

```
rocommunity=COMMUNITY-STRING
```

Angabe des Community Strings, der zur Abfrage der SNMP-Agenten der einzelnen Endgeräte dient.

Zusätzlich sind optionale Parameter möglich bzw. für manche Tests nötig:

```
masterswitch=IP-ADRESSE
```

Angabe der IP-Adresse des Switches, bei dem der Location Test (vgl. 5.5.14) beginnen soll, d.h. in einer Umgebung mit kaskadierten Switches derjenige, der sich in der obersten Ebene der Kaskade befindet. Ist dieser Parameter nicht angegeben, wird der Location Test kein Ergebnis liefern.

Der letzte Abschnitt der Konfigurationsdatei erlaubt die Umsetzung des in 2.2 beschriebenen Ansatzes der verschiedenen Ansichten auf die gesammelten Daten. Es werden sog. Views definiert, die es erlauben gezielt einzelne Informationen zu gruppieren und gruppiert darzustellen:

```
view {  
  name=NAME  
  hostsfile=FILENAME  
  update-interval=MINUTES  
  descr=DESCRIPTION STRING  
}
```

Die zu einem View gehörigen Parameter sind in geschweifte Klammern eingefasst. Es müssen alle vier Parameter angegeben sein, wobei der Name und die Beschreibung frei zu wählen sind. *hostsfile* gibt an, in welcher Datei die weitere Konfiguration des Views zu finden ist. Eine Beschreibung der Datei folgt im nächsten Abschnitt. *update-interval* gibt an in welchen Zeitabschnitten die Tests wiederholt werden sollen. Die zentrale Konfiguration muss mindestens einen View enthalten. Die maximale Anzahl der Views ist lediglich durch die Systemressourcen des Überwachungsrechners beschränkt.

Für jeden View wird ein eigener Thread gestartet, dem die angegebenen Parameter übergeben werden.

Eine Hostsdatei definiert, welche Endgeräte überwacht werden sollen; genauer, welche Dienste auf den entsprechenden Geräten. Zusätzlich ist eine Einteilung in Gruppen vorgesehen, um die Präsentation der Daten übersichtlicher gestalten zu können. Der Aufbau der Konfigurationsdatei ähnelt der hosts-Datei auf Linux/Unix Systemen:

```
group GRUPPENNAME  
IP-ADRESSE NAME # DIENST1 DIENST2
```

- oder anschaulicher -

```
group Linux-Servers  
10.10.1.1 www.mydomain.local # http pop3
```

Beim Auswerten der Hostsdatei erfolgt für jede Gruppe ein Eintrag in einem Hash mit dem Namen der Gruppe als Schlüssel und einer Referenz auf eine Liste als Wert. Für jedes überwachte Endgerät einer Gruppe erfolgt nun ein Eintrag in diese Liste mit einer Referenz auf ein Objekt vom Typ Host. Die Attribute eines Host-Objekts sind neben dem Namen und der IP-Adresse noch eine Liste mit den Namen der durchzuführenden Tests.

### 5.4.2 Durchführen der Tests

Nach Aufbau dieser Struktur beginnt das Programm mit der Iteration durch alle Gruppen und den darin enthaltenen Hostobjekten. Auf jedes Hostobjekt wird eine Methode aufgerufen, die zunächst die Erreichbarkeit des Zielhosts mittels eines einfachen Ping prüft. Ist der Zielhost nicht erreichbar, werden die gewünschten Tests gar nicht erst durchgeführt.

Bei positiver Erreichbarkeit wird zunächst das Vorhandensein eines SNMP-Agenten auf dem zu überwachenden Host geprüft. Dies geschieht durch die Abfrage der OID *iso.org.dod.internet.mib-2.system.sysDescr*, da die System MIB, laut RFC1213, in allen SNMP-Agenten implementiert sein muss. Sollte das Ergebnis dieses Tests negativ, also kein Agent auf dem Zielhost vorhanden sein, so werden nur die „realen Tests“ durchgeführt. Für die SNMP-basierten Tests wird vermerkt, dass sie auf Grund des Fehlens oder der falschen Konfiguration (fehlende Zugriffsrechte des Managers) des Agenten nicht durchgeführt wurden.

### 5.4.3 Die Ergebnisse

Um die Einsatzmöglichkeiten so wenig wie möglich einzugrenzen, sind Testergebnisse nicht auf einige wenige diskrete Statuswerte begrenzt, sondern frei wählbar. Dennoch gibt es eine Reihe von vordefinierten Testergebnissen. Diese dienen hauptsächlich der Übersichtlichkeit und sind deshalb als Farben definiert:

- grün: Test erfolgreich / alles in Ordnung
- gelb: Achtung / es könnte Probleme geben
- rot: Problem
- violett: Kein Ergebnis
- farblos: Kein Ergebnis, aber das ist in Ordnung

Es sind aber beliebige andere Werte (Zahlen oder Strings) als Ergebnis zugelassen, was besonders deshalb wichtig ist, weil auch rein informative, aber nicht systemkritische Tests Bestandteil des Gesamtprojekts sind. Um dem aufrufenden Programm mitzuteilen, ob es sich um eines der vordefinierten Ergebnisse oder ein beliebiges anderes Ergebnis handelt, werden die Farbangaben mit dem '&'-Zeichen als Präfix versehen. Die Menge der vordefinierten Ergebnisse ist also {&green, &yellow, &red, &purple, &clear}.

Die Ergebnisse jedes Tests werden in einem Statusobjekt gespeichert, dessen Attribute alle gewünschten Informationen enthalten (Testname, Ergebnis, Datum/Uhrzeit, Anmerkungen). Die Statusobjekte der Ergebnisse aller Tests eines Hosts sind von einer Hashtabelle, die ein Attribut des Hostobjekts ist, referenziert und werden erst beim nächsten Durchlauf aller Tests überschrieben.

#### 5.4.4 Aufbereitung der Ergebnisse

Nachdem alle Tests beendet sind, werden zunächst die Statusseiten erzeugt, die die Ergebnisse präsentieren. Alle Ergebnisse werden in Kurzfassung auf einer einzigen Seite zusammengefasst (s. Abb. 5.1). Hierbei wird nur das eigentliche Testergebnis berücksichtigt, evtl. Anmerkungen werden auf dieser Seite nicht angezeigt.

Zum Aufbau dieser Seite wird für jede Gruppe zunächst einmal bestimmt, wie viele und welche Testergebnisse dargestellt werden müssen. Dazu werden aus jedem Hostobjekt die Schlüssel der Hashes, in denen die Referenzen auf die einzelnen Testergebnisse stehen, ausgelesen und in einem Array zusammengefasst - wobei doppelte Einträge vermieden werden. Das Ergebnis ist ein Array mit den Namen aller Tests der Gruppe.

Mit diesen Informationen wird eine Tabelle mit einer Spalte für jeden Test und einer Zeile für jeden Host der Gruppe erstellt. Die Tabellenzellen enthalten das Testergebnis, dargestellt durch eine Grafik für die vordefinierten

Ergebniswerte oder den Wert selber für die anderen. Desweiteren ist jede Ergebnisdarstellung ein Link zu einer weiteren Seite mit Details zu dem Testergebnis, die im folgenden Schritt erstellt werden.

Nach Erstellen der Zusammenfassung wird für jeden durchgeführten Test eine Seite erstellt, in der Details zum Test zu finden sind (s. Abb. 5.2).

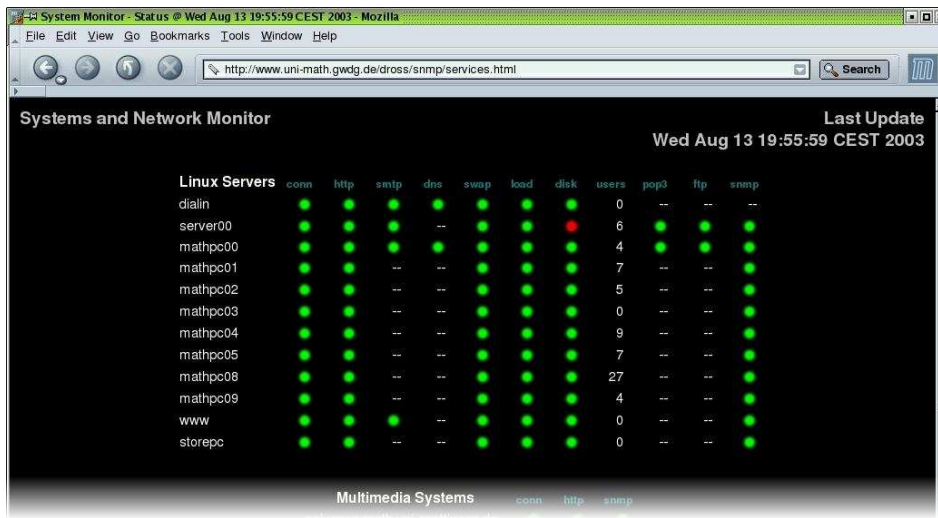


Abbildung 5.1: Screenshot einer Statusübersichtsseite

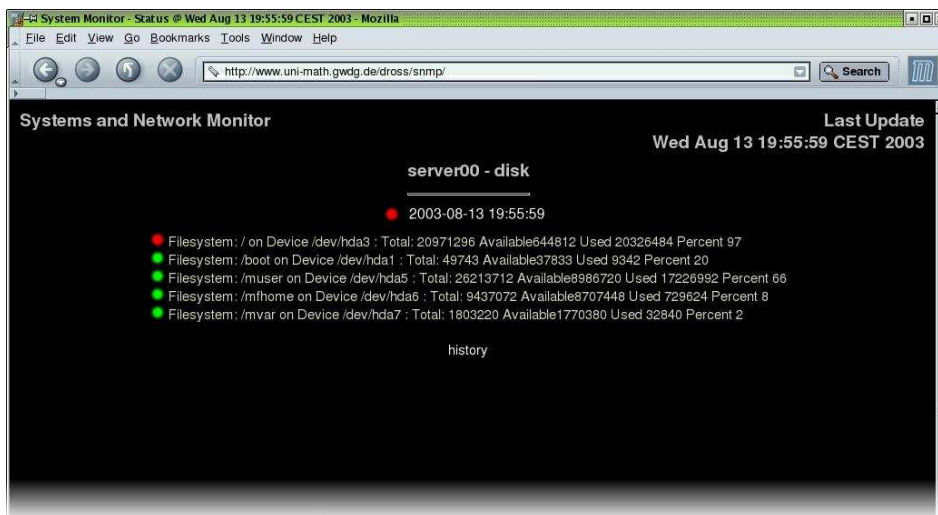


Abbildung 5.2: Screenshot einer Ergebnisdetailsseite

### 5.4.5 Sicherung der Ergebnisse

Eine weitere Iteration über die Gruppen und darin enthaltenen Hosts liest die Ergebnisse abermals aus und vergleicht sie mit den Ergebnissen des vorherigen Durchlaufs in der Datenbank. Bei Abweichungen werden die neuen Ergebnisse in der Datenbank gesichert. Eine detaillierte Beschreibung des Datenbanklayouts folgt im Abschnitt über das Datenbanklayout (5.8).

### 5.4.6 Wiederholung

Nach einem vorgegebenen Zeitintervall beginnen die Tests, die Aufbereitung der Ergebnisse und die Sicherung von Neuem, solange bis das Programm abgebrochen wird.

## 5.5 Die Tests

Die zur Verfügung stehenden Tests entscheiden maßgeblich, inwieweit der Einsatz eines solchen Werkzeugs die Arbeit erleichtert. In vielen Netzwerken findet man Dienste, die in anderen eher selten vorkommen. Eine Implementation, die alle Eventualitäten abdeckt, ist nicht möglich. Daher wurden nur Tests der Dienste implementiert, die in den meisten Netzen vorkommen und eine Schnittstelle für weitere Tests geschaffen (siehe 5.7). Die implementierten Tests werden im Folgenden kurz beschrieben:

### 5.5.1 Erreichbarkeit (conn)

Dieser Test wird vor allen anderen ausgeführt und nicht in der Konfigurationsdatei angegeben. Ein Ping prüft die Erreichbarkeit des zu testenden Hosts, d.h., es wird ein ICMP Echo Request gesendet, der i.d.R. einen ICMP Echo Response des gepingten Endgerätes auslöst, wenn dieses erreichbar ist. Zum Pingen des Hosts werden nicht die Funktionen genutzt, die das Perl Modul Net::Ping liefert, da mit diesen ein ICMP-Ping nur möglich ist, wenn das Perl-Skript mit root-Rechten ausgeführt wird. Stattdessen wird ein Systemaufruf gemacht und dessen Rückgabewert ausgewertet. Der Test liefert eines von drei möglichen Ergebnissen:

- &green, wenn das zu testende Gerät auf den Echo Request reagiert hat
- &clear, wenn das Gerät nicht reagiert hat, aber in der Konfigurationsdatei angegeben wurde, dass dies in Ordnung ist (dialup)
- &red, wenn das Gerät nicht reagiert hat

### 5.5.2 Webserver (http)

Der Test stellt eine einfache Anfrage an einen HTTP-Server und wertet die Header der Antwort aus. Mögliche Ergebnisse des Tests sind:

- &green, wenn der HTTP Header der Antwort den Status Code 2xx (Success) enthält
- &red, wenn der HTTP Header einen Fehler enthält, oder der Host nicht erreichbar ist

### 5.5.3 FTP Server (ftp), SMTP Server (smtp), POP3 Server (pop)

Diese Tests prüfen die Verfügbarkeit der jeweiligen Dienste durch den Versuch eines Verbindungsaufbaus. Mögliche Ergebnisse dieser Tests sind:

- &green, wenn der Verbindungsaufbau gelingt
- &red, wenn der Verbindungsaufbau fehlschlägt

### 5.5.4 Domain Name Service (dns)

Der Test prüft die Verfügbarkeit und Funktionalität eines Domain Name Servers, indem er eine Anfrage an den Server stellt und die Antwort auswertet. Mögliche Ergebnisse sind:

- &green, wenn die Namensauflösung gelingt
- &red, wenn die Namensauflösung fehlschlägt

### 5.5.5 Secure Shell (ssh)

Der Test versucht eine Verbindung zum SSH Daemon des zu überwachenden Hosts aufzubauen und prüft damit dessen Verfügbarkeit. Mögliche Testergebnisse sind:

- &green, wenn der SSH Daemon verfügbar ist,
- &red, wenn der SSH Deamon nicht verfügbar ist.

### 5.5.6 SNMP-Agent (snmp)

Der Test prüft die Verfügbarkeit eines SNMP-Agenten auf den zu überwachenden Host, indem eine Anfrage nach der OID *system.sysDescr* gemacht wird. Der Test wird durchgeführt, unabhängig davon, ob er in der

Konfigurationsdatei angegeben ist; die Angabe in der Konfiguration beeinflusst allerdings, ob das Ergebnis dieses Tests präsentiert wird oder nicht. Das Bestehen dieses Tests ist Voraussetzung dafür, dass die SNMP-basierten Tests überhaupt durchgeführt werden. Mögliche Ergebnisse sind:

- &green, wenn der SNMP Get-Request gelingt
- &red, wenn der SNMP Get-Request fehlschlägt

### 5.5.7 Swapspace (swap)

Der Test fragt den SNMP-Agenten des zu überwachenden Hosts nach den Werten für die OIDs *enterprises.ucdavis.memory.memTotalSwap* und *enterprises.ucdavis.memory.memAvailSwap* und berechnet daraus den Prozentsatz des freien Swap-Speichers. Mögliche Rückgabewerte:

- &green, wenn der freie Swap-Speicher über 10% liegt
- &yellow, wenn der Swap-Speicher zwischen 5% und 10% liegt
- &red, wenn der freie Swap-Speicher unter 5% liegt

### 5.5.8 Festplattenplatz (disk)

Der Test überprüft den zur Verfügung stehenden Platz auf den Festplatten des zu überwachenden Hosts anhand der Einträge in der Tabelle *enterprises.ucdavis.dskTable*. Wie in Abschnitt 5.3 (Konfiguration von NetSNMP) beschrieben, muss dazu für jede zu überwachende Partition ein Eintrag in der Konfigurationsdatei des Agenten gemacht werden. Mögliche Testergebnisse:

- &green, wenn der freie Platz auf allen überwachten Partitionen größer als 10% ist
- &yellow, wenn der freie Platz auf einer oder mehreren der überwachten Partitionen zwischen 5 und 10% liegt
- &red, wenn der freie Platz auf einer oder mehreren der überwachten Partitionen weniger als 5% beträgt

### 5.5.9 Prozessorlast (load)

Der Test liefert den durchschnittlichen CPU-Load der letzten Minute. Der Wert wird durch die OID *enterprises.ucdavis.laTable.laEntry.laLoad.1* ermittelt. Mögliche Rückgabewerte sind:

- &green, wenn der Load kleiner als 2,5 ist
- &yellow, wenn der Load zwischen 2,5 und 4 liegt
- &red, wenn der Load größer als 4 ist

### 5.5.10 Benutzerzahl (users)

Der SNMP-basierte Test liefert die Anzahl der Benutzer des angegebenen Hosts. Die Ermittlung des Ergebnisses erfolgt durch Abfrage der OID *host.hrSystem.hrSystemNumUsers*. Mögliche Ergebnisse sind:

- Anzahl der Nutzer auf dem System
- &yellow, wenn der gefragte SNMP-Agent die host MIB nicht implementiert hat oder der Zugriff verweigert wurde
- &red, wenn der SNMP-Agent nicht antwortet

### 5.5.11 XHost (xhost)

Der Test versucht eine eindeutige Bestimmung des Rechners, auf dem der Benutzer eines X-Terminals arbeitet. Dazu muss ein Skript auf dem X-Terminal installiert sein, welches der NetSNMP-Agent aufruft. Das Skript analysiert die TCP-Verbindungen des Rechners; insbesondere die, die dem X-Server zugeordnet sind. Mögliche Ergebnisse:

- Name der Workstation, auf dem der Benutzer des Terminals arbeitet
- &purple, wenn der Rechner nicht ermittelt werden kann

### 5.5.12 Prozessorinformation (cpuinfo)

Der Test gibt Informationen zum installierten Prozessor an; dazu muss auf dem zu testenden Rechner ein Skript installiert sein, welches der NetSNMP-Agent ausführen kann. Dieses Skript liest die notwendigen Informationen aus */proc/cpuinfo*. Möglicher Ergebnisse sind:

- Beschreibung des Prozessors
- &purple, wenn die Information nicht ermittelt werden kann

### 5.5.13 Speicherinformationen (meminfo)

Der Test ermittelt die Größe des installierten Speichers über die Abfrage der Variablen *host.hrStorage.hrMemorySize (.1.3.6.1.2.1.25.2.2)* des SNMP-Agenten auf dem zu überwachenden Host:

- Größe des installierten Speichers in KiloByte
- &purple, wenn die Information nicht ermittelt werden kann

#### 5.5.14 Aufstellungsort (location)

Wie bereits in der Einleitung kurz erwähnt, versucht dieser Test anhand der ihm zur Verfügung stehenden Daten zu ermitteln, wo sich das Gerät mit der angegebenen IP-Adresse verbindet. Voraussetzungen für diesen Test: Alle Switches müssen SNMP-fähig sein und die *bridge*-MIB implementieren. Es muss eine Zuordnung von Switch Interfaces zu Räumen (Büros) existieren. Da Switches nicht auf IP-Protokollebene arbeiten, muss ein Weg gefunden werden, um die Hardware Adresse (MAC-Adresse) zu einer gegebenen IP-Adresse heraus zu finden. Diese Funktionalität ist in einem Modul gekapselt, so dass evtl. betriebssystemabhängige, notwendige Anpassungen erleichtert werden (Siehe auch 5.9). In der *bridge*-MIB findet man unter *bridge.dot1dTp.dpt1dTpFdbTable* eine Tabelle, in der die Forwarding Database des Switches enthalten ist. Die erste Spalte jedes Tabelleneintrags enthält eine MAC-Adresse und die zweite Spalte die Portnummer, an die Pakete für diese MAC-Adresse gesendet werden.

Diese Portnummer ist ebenfalls in der Tabelle *bridge.dot1dBase.dot1dBasePortTable* enthalten, in der auch ein Eintrag mit einem Verweis auf den entsprechenden *ifIndex* aus der bereits bekannten *interfaces.ifTable* ist. Auf diese Weise lässt sich anhand der Hardware-Adresse eines Geräts feststellen, an welchem Interface eines Switches es angeschlossen ist. Befindet man sich in einer Umgebung, in der nur ein Switch (oder mehrere zu einem Stack zusammengeschaltete) vorhanden ist, so kann mit einer einfachen Zuordnung von Interface zu Büronummer auch direkt der Standort eines Gerätes ermittelt werden. In einer Umgebung mit kaskadierten Switches ist dies nicht so einfach, da hier noch weitere Anfragen andere Switches nötig sein können. Unter anderem um diese Fälle abzudecken, ist die "Switch Interface / Raum" bzw. "Switch Interface / Switch" Zuordnung komplett in einer Datenbank abgelegt. Mit Hilfe eines Webinterfaces für die Datenbank wird die Erstellung und Pflege einer solchen Zuordnung erleichtert. Anhang B beschreibt das dazugehörige Datenbanklayout und das Webinterface. Mögliche Ergebnisse des Tests:

- Nummer des Raumes, in der sich das Gerät befindet und evtl. weitere Details zu dem Raum (z.B. Telefonnummer oder Ansprechpartner)
- &purple, wenn die Raumnummer nicht ermittelt werden konnte

### 5.5.15 Uptime (uptime)

Der Test erfragt die Zeit, die der Host schon aktiv ist (uptime), also die Zeit seit dem letzten Systemstart. Die Ermittlung des Ergebnisses erfolgt über die OID *host.hrSystem.hrSystemUptime*. Mögliche Ergebnisse sind:

- Zeit seit dem letzten Systemstart (angegeben in Tagen, Stunden und Minuten)
- &yellow, wenn der gefragte SNMP-Agent die *host*-MIB nicht implementiert hat oder der Zugriff verweigert wurde
- &red, wenn der SNMP-Agent nicht antwortet

## 5.6 Verwendete Perl Module

Alle verwendeten Perl Module sollten Bestandteil der meisten größeren Linux Distributionen sein; oftmals sind sie aber nicht Bestandteil der Standardinstallation.

### 5.6.1 Net::SNMP

Das Net::SNMP Modul ist eine SNMP-Bibliothek zum Absetzen von SNMP-Requests. Es wird in allen SNMP-basierten Tests benutzt. Weitere Informationen zu diesem Modul sind unter <sup>1</sup> verfügbar.

### 5.6.2 Tie:IxHash

Die Gruppen werden wie beschrieben in Hashes untergebracht. Allerdings kommt hier nicht der "normale" Perl-Hash zum Einsatz, sondern eine Erweiterung namens IxHash. Der Vorteil von IxHash ist, dass die Reihenfolge der eingefügten Elemente erhalten bleibt. Dadurch wird erreicht, dass die Anordnung der Gruppen bei der Präsentation der Ergebnisse der Anordnung in der Konfigurationsdatei entspricht.

### 5.6.3 Net::DNS

Das Net::DNS Modul stellt eine Resolver Schnittstelle zu einem Domain Name System (DNS) bereit, und erlaubt DNS Anfragen in Perl. Der DNS-Test benutzt das Modul für eine Anfrage nach der IP-Adresse von [www.perl.com](http://www.perl.com).

---

<sup>1</sup><http://www.cpan.org>

### 5.6.4 Net::POP3

Das Net::POP3 Modul implementiert ein Client Interface zum Post Office Protocol Version 3 und wird benutzt, um einen Verbindungsaufbau zu einem POP-Server zu testen.

### 5.6.5 Net::SMTP

Das Net::SMTP Modul stellt ein Interface zur Verfügung, um Verbindungen zu Simple Message Transfer Protocol Servern aufzubauen. Der SMTP-Test basiert auf diesem Modul.

### 5.6.6 Net::FTP

Net::FTP ist eine Klasse, die einen einfachen FTP Client in Perl implementiert und für den FTP-Test benutzt wird.

## 5.7 Erweiterbarkeit

Die Erweiterbarkeit eines solchen Überwachungssystems spielt - wie bereits erwähnt - für die Nutzbarkeit eine sehr große Rolle, denn in jedem Netz lassen sich Endgeräte oder Dienste finden, die in anderen vielleicht nicht vorkommen. Eine Sammlung von Tests vorzugeben, die alle Eventualitäten abdeckt, ist daher nicht möglich. Daraus resultiert ein System, welches auf einfache Art und Weise erweitert werden kann.

### 5.7.1 Anforderungen an die Erweiterbarkeit

Um die Anpassung an die lokalen Begebenheiten so einfach wie möglich zu machen, werden folgende Anforderungen an die Erweiterbarkeit des Systems gestellt:

- Das System sollte ohne Änderungen am bestehenden Code erweitert werden können.
- Die Anforderungen und die Restriktionen an ein Erweiterungsmodul sollten so gering wie möglich sein.

- Die Schnittstelle zwischen dem System und den Erweiterungen sollte so einfach wie möglich sein.
- Die Wahl der Implementationssprache für Erweiterungen sollte dem Benutzer überlassen werden.

### 5.7.2 Umsetzung der Erweiterbarkeit

Um den ersten der geforderten Punkte zu erfüllen, muss zunächst ein Weg gefunden werden, dass das System ein Modul aufrufen kann, von dessen Existenz es nichts weiß. Dieser Punkt, und die Forderung nach der freien Wahl der Implementationssprache, führen dazu, dass es sich bei den Erweiterungen um ausführbare Programme handeln muss, die vom Überwachungssystem durch einen Systemaufruf gestartet werden.

In der hosts-Datei wird dann lediglich der Programmname angegeben, den das System aufrufen soll. Damit das System die ausführbare Datei findet, muss sie sich im Verzeichnis *ext* unterhalb des Hauptverzeichnisses des Überwachungswerkzeuges befinden.

Als formale Schnittstelle sind zwei Dinge zu definieren:

- die Schnittstelle zum Aufruf des Tests und
- die Schnittstelle zur Übergabe der Testergebnisse.

Da es sich bei den Erweiterungen um ausführbare Programme handelt, ist die Schnittstelle zum Aufruf recht einfach: Sie besteht aus den Kommandozeilenparametern an das Programm. Auch die übergebenen Informationen beschränken sich auf ein Minimum, nämlich auf die IP-Adresse des zu testenden Geräts und den Community String des SNMP-Agenten.

Die Schnittstelle zur Rückgabe der Testergebnisse hingegen ist aufgrund des Informationsgehaltes nicht ganz so einfach. Der erste, naheliegende Gedanke, der Rückgabewert des Programmes, entfällt daher leider als Möglichkeit. Auch die Option, temporäre Dateien oder Sockets zur Übertragung der Ergebnisse zu nutzen, scheitert an den Anforderungen, da sie die Implementation einer Erweiterung unnötig verkomplizieren würde und unter Umständen die Einsetzbarkeit des Systems auf unterschiedlichen Plattformen unmöglich machen könnte. Letztlich wurde die Standardausgabe zur Übergabe der Testergebnisse gewählt, da sie völlig unabhängig von der Implementationssprache ist und das Ausgeben von Daten auf die Standardausgabe in jeder Sprache eine der einfachsten Aufgaben ist.

Diese Entscheidung bringt auch Nachteile mit sich: Bei den so übermittelten Daten handelt es sich um reine Textdaten ohne jegliche weitere Informationen zum Datentyp. Daher muss den Daten eine klar definierte Struktur zugrunde gelegt werden, damit das Überwachungssystem die Daten richtig interpretieren kann.

Betrachtet man ein Testergebnis genauer, so handelt es sich um vier Einzelinformationen, die einen Status beschreiben: Der Name des Tests, das Ergebnis des Tests (nach der Definition in 5.4.3), das genaue Datum bzw. die Uhrzeit des Tests und weitere Anmerkungen oder Details.

Bei den ersten drei dieser Informationen handelt es sich um kurze, einzeilige Informationen; lediglich über die Anmerkungen kann man keinerlei Aussagen machen. Als Trennzeichen zwischen den einzelnen Informationen bietet sich daher ein Zeilenende(`\n`) an. Das Format für ein Testergebnis sieht demnach folgendermaßen aus: "Testergebnis\nTestname\nDatum\n evtl. mehrzeiliges Testergebnis".

Die an die Erweiterungsmöglichkeiten gestellten Anforderungen legen aber noch eine Ergänzung zu diesem Format nahe. Es ist durchaus denkbar, dass ein aufgerufenes Programm mehrere Tests gleichzeitig durchführt und somit auch mehrere Ergebnisse zurückliefern können soll. Deshalb wird ein weiteres Trennzeichen eingeführt: Zeilenende.Zeilenende (`\n.\n`), also ein einzelner Punkt auf einer Zeile (in Anlehnung an das Datenformat für das Simple Message Transfer Protocol). Das endgültige Format für die ErgebnISRückgabe sieht also wie folgt aus: "Testergebnis1\nTestname1\nDatum1\nAnmerkung1\n.\n Testergebnis2" usw.

## 5.8 Datenbank und Datenbanklayout

Die Datenbank wird genutzt, um einzelne Testergebnisse zu speichern, wodurch ein zeitlicher Verlauf der Zustandsdaten darstellbar ist. Es werden nicht alle Testergebnisse in der Datenbank gespeichert, sondern nur solche, bei denen sich das Testergebnis im Gegensatz zum vorherigen geändert hat. Die Einträge in der Datenbank lassen sich also als Ereignisse auffassen bzw. als Zeitpunkte einer Zustandsänderung. Abb. 5.3 zeigt das Entity-Relation Modell der Datenbank, in der die Zustandsänderungen gespeichert werden. Die Datenbank wurde als Speichermedium einfachen Log-Dateien vorgezogen, weil dadurch eine spätere Nutzung der Daten vereinfacht wird. Weitere Module könnten anhand der in der Datenbank gespeicherten Zustandsänderungen beispielweise statistische Analysen über die Verfügbarkeit einzelner Dienste erstellen.

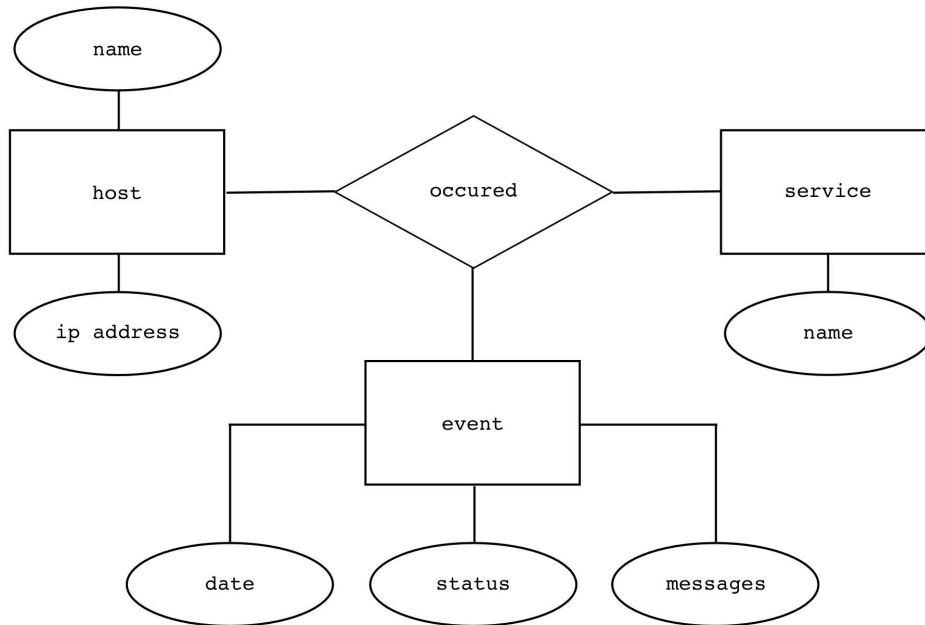


Abbildung 5.3: Entity Relation Modell der Datenbank zum Speichern der Zustandsänderungen

## 5.9 Probleme und deren Lösung

Bei der Implementation ergaben sich zwei Probleme, die hier stellvertretend für möglicherweise bei eigenen Implementationen oder Erweiterungen auftretenden Problemen kurz erläutert werden sollen.

### 5.9.1 Ping

In einigen Netzen werden ICMP-Requests aus Sicherheitsgründen nicht beantwortet, was dazu führt, dass das Netzwerküberwachungssystem das Endgerät für nicht erreichbar hält und somit gar nicht mit den eigentlichen Tests beginnt. Ping wurde aber zu genau diesem Zweck eingeführt und wird daher in diesem Zusammenhang auch so genutzt. Eine Änderung dieses Ansatzes ist zwar möglich, aber nicht zu empfehlen, da das Testen von nicht aktiven Endgeräten sehr lange dauern würde, da bei jedem Test ein Timeout abgewartet wird bevor der Test eines Dienstes einen Fehler meldet.

### 5.9.2 Bestimmung der Hardware Adresse zu einer IP Adresse

Für den “location”-Test ist es zwingend notwendig die Hardware-Adresse (MAC Adresse) eines Netzwerkdevices zu einer vorgegebenen IP-Adresse in Erfahrung zu bringen. Diese Aufgabe ist nicht so leicht, wie sie auf den ersten Blick scheint. Die Kernel moderner Betriebssysteme halten zu diesem Zweck sogenannte ARP (Address Resolution Protocol) Tabellen vor, in denen alle dem System bekannten IP-Adressen aus dem eigenen Netz vorhanden sein *sollten*. Das Nachschlagen einer IP-Adresse in einer solchen Tabelle ist auf jedem Betriebssystem mit Unterstützung für IP-basierte Netze ähnlich einfach. Es treten jedoch Situationen auf, in denen ein Nachschlagen in dieser Tabelle nicht das gewünschte Ergebnis liefert. Linux Kernel z.B. haben keinen Eintrag für die eigene IP-Adresse in dieser Tabelle, während das auf Solaris Systemen sehr wohl der Fall ist. Der Programmcode zur Bestimmung einer MAC-Adresse zu einer gegebenen Hardware Adresse wurde daher komplett in ein eigenes Modul ausgelagert und gekapselt, sodass Anpassungen für den Einsatz auf anderen Betriebssystemen erleichtert werden.

# Kapitel 6

## Voraussetzungen und Installation

### 6.1 Voraussetzungen

Die Voraussetzungen zur Benutzung der im Rahmen dieser Arbeit entwickelten Beispielimplementation einer Netzwerküberwachungs- und Informationssoftware ergeben sich aus den im Kapitel 4 dargestellten Komponenten. Benötigt werden eine Workstation, die als SNMP-Manager fungiert und die eigentliche Überwachung und Zusammenstellung der Daten übernimmt, sowie eine auf dieser Workstation lauffähige Perl Umgebung (Version 5.6.1 oder höher) inkl. der in Kapitel 5.6 beschriebenen Perl Module und eine vom Perl DBI Modul und von PHP unterstützte Datenbank zum Ablegen der Zustandsänderungen.

Zur Präsentation der Daten muss ein Webserver mit PHP Unterstützung vorhanden sein. Um Workstations im gewünschten Umfang überwachen zu können, muss auf diesen ein SNMP-Agent installiert und so konfiguriert sein, dass der Management-Host Zugriff auf die Daten des Agenten hat. Einige der implementierten Tests setzen NetSNMP als SNMP-Agent voraus und verlangen das Vorhandensein bestimmter Skripte auf dem zu überwachenden Host (vgl. 5.5).

### 6.2 Installation

Die Installation und der damit verbundene Arbeitsumfang ist von der Größe des zu überwachenden Netzes bzw. von der Anzahl der zu überwachenden Dienste abhängig. Nach Erstellung der in Abschnitt 5.4.1 beschriebenen Konfigurationsdatei und der ebenfalls in dem Abschnitt beschriebenen Hostsdateien muss einmal das enthaltene Installations-Skript

aufgerufen werden, welches die benötigten Tabellen in der Datenbank anlegt. Nach dem erstmaligen Start des Systems steht dann das Webinterface zur Administration der Datenbank zur Verfügung (vgl. Anhang B).

# Kapitel 7

## Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde eine Beispielimplementierung eines Netzwerküberwachungs- und Informationssystems entwickelt, anhand dessen die Einsatzmöglichkeiten eines solchen Systems und die daraus resultierenden Vorteile dargestellt wurden. Der Schwerpunkt lag in der Beschreibung des Simple Network Management Protocols und dessen Einsatz zum Sammeln der Zustandsdaten von Netzwerkkomponenten.

Neben einer detaillierten Beschreibung des Simple Network Management Protocols wurde die Konfiguration von SNMP-Komponenten anhand der auf Linux Systemen am weitesten verbreiteten SNMP-Implementation erläutert.

Das Simple Network Management Protocol ist das einzige standardisierte Protokoll, mit dem sich Überwachungs- und Managementsysteme effizient in Netzwerken realisieren lassen; wobei im Rahmen dieser Arbeit die Überwachungsaufgaben im Vordergrund standen und die Management Funktionen von SNMP nur eine Nebenrolle spielten.

Anhand der Beispielimplementierung ist ebenfalls verdeutlicht worden, dass das Überwachen der Verfügbarkeit einzelner Dienste und das Sammeln von Informationen über die im Netzwerk vorhandene Hardware von technischer Seite gesehen dieselbe Aufgabe ist und sich mit Hilfe desselben Werkzeuges erledigen lässt.

Weiterhin wurde verdeutlicht, dass ein solches oder ähnliches System in Netzwerken, bei denen die Verfügbarkeit einzelner Dienste von essentieller Bedeutung ist, ab einer gewissen Größe des Netzes unverzichtbar ist. Der Einsatz menschlicher Kapazitäten zur Bewältigung der Aufgaben, die ein derartiges Überwachungssystem erfüllt, ist in dieser Form und in solchem Umfang nicht möglich.

Die hier diskutierte Implementation ist in einem begrenzten Zeitraum von ca. drei Monaten entstanden und obwohl sie sich in der Praxis bereits bewährt hat, beschränkt sie sich auf die wesentlichen Aspekte. Das entwickelte System ist zwar so konzeptioniert, dass es ohne Änderungen am bestehenden Code erweitert werden kann, diese Erweiterbarkeit beschränkt sich jedoch auf das Überwachen zusätzlicher Dienste bzw. das Sammeln zusätzlicher Informationen. Unter Umständen kann für den praktischen Einsatz, je nach Umgebung, aber eine Erweiterung der Funktionalität notwendig sein. Anhang E beinhaltet einige Anregungen zur Erweiterung der Funktionalität.

Abschließend betrachtet lässt sich sagen, dass Systeme wie das hier beispielhaft entwickelte in der Zukunft im Umfeld der Administration immer mehr an Bedeutung gewinnen werden, da die eingesetzten Netze immer größere und komplexere Dimensionen annehmen werden.

# Anhang A

## RFCs

Im Folgenden sind die wichtigsten der für den Umgang mit SNMP relevanten RFCs thematisch gruppiert aufgelistet:

### A.1 SNMPv1 Protokoll

| <b>RFC</b> | <b>Beschreibung</b>                | <b>Status</b> |
|------------|------------------------------------|---------------|
| 1157       | Simple Network Management Protocol | Full          |
| 1418       | SNMP over OSI                      | Proposed      |
| 1419       | SNMP over Apple Talk               | Proposed      |
| 1420       | SNMP over IPX                      | Proposed      |

### A.2 SNMPv2 Protokoll

| <b>RFC</b> | <b>Beschreibung</b>            | <b>Status</b> |
|------------|--------------------------------|---------------|
| 1901       | Community-based SNMPv2         | Experimental  |
| 1905       | Protocol Operations for SNMPv2 | Draft         |
| 1906       | Transport Mappings for SNMPv2  | Draft         |
| 1907       | MIB for SNMPv2                 | Draft         |
| 1909       | Administrative Infrastructure  | Experimental  |
| 1910       | User-based Security Model      | Experimental  |

## A.3 SNMPv3 Protokoll

| <b>RFC</b> | <b>Beschreibung</b>                | <b>Status</b> |
|------------|------------------------------------|---------------|
| 2570       | Introduction to SNMPv3             | Informational |
| 2571       | Architecture for SNMP Frameworks   | Draft         |
| 2572       | Message Processing and Dispatching | Draft         |
| 2573       | SNMP Applications                  | Draft         |
| 2574       | User-based Security Model          | Draft         |
| 2575       | View-based Access Control Model    | Draft         |
| 2576       | Coexistence between SNMP Versions  | Proposed      |

## A.4 SMIV1 Data Definition Language

| <b>RFC</b> | <b>Beschreibung</b>                 | <b>Status</b> |
|------------|-------------------------------------|---------------|
| 1155       | Structure of Management Information | Full          |
| 1212       | Concise MIB Definitions             | Full          |
| 1215       | A Convention for Defining Traps     | Informational |

## A.5 SMIV2 Data Definition Language

| <b>RFC</b> | <b>Beschreibung</b>                 | <b>Status</b> |
|------------|-------------------------------------|---------------|
| 2578       | Structure of Management Information | Full          |
| 2579       | Textual Conventions                 | Full          |
| 2580       | Conformance Statements              | Full          |

# Anhang B

## Zuordnung von Switch Interfaces zu Räumen

Die Zuordnung von Switch Interfaces zu Räumen, die zur Lokalisierung der vorhandenen Netzwerkkomponenten genutzt wird (vgl. 5.5.14) befindet sich vollständig in einer Datenbank. Der Hauptgrund dafür ist, dass dies der einzige Punkt in dem System ist, an dem evtl. manuelles Eingreifen notwendig ist, damit das System die gewünschten Ergebnisse liefert. Ein solches Eingreifen wird genau dann notwendig, wenn sich die physikalische Verkabelung der Switches ändert. Das Datenbank-Layout ist den tatsächlichen Begebenheiten, die üblicherweise vorzufinden sind, nachempfunden, um die Pflege der Daten so einfach wie möglich zu machen.

### B.1 Datenbanklayout

Das Entity-Relation Modell (s. Abb. B.1) zeigt eine Relation *Patchpanel*, die in der Datenbank als eigene Tabelle abgebildet wird. Diese Tabelle enthält zwei Spalten: die Nummer der Dose im Patchfeld als Primärschlüssel und einen Verweis auf einen Eintrag in der Raum-Tabelle. Diese Tabelle wird in der Regel in den meisten Umgebungen einmal eingerichtet und muss dann nicht mehr verändert werden. Die *Switch Interface Tabelle* enthält einen Eintrag für jedes Interface der vorhandenen Switches und jeder dieser Einträge besitzt einen Verweis auf einen Eintrag in der *Patchpanel* Tabelle. Dieser Verweis ist die Abbildung des realen Kabels von einem Switch Interface zur eine Dose im Patchfeld.

Ein Switch Interface besitzt insgesamt vier Attribute (vgl. Abb. B.1): Die Nummer des Interfaces dient der eindeutigen Identifikation und muss mit

der intern von Switch verwendeten Nummer übereinstimmen, die über die *interfaces.ifTable* Tabelle des SNMP-Agenten des Switches als Index abrufbar ist (vgl. 5.5.14). Die Beschreibung dient nur der Erleichterung der Pflege der Daten; sie ist frei wählbar und wird systemintern nicht verwendet.

Das *patchpanel port* Attribut enthält den bereits erwähnten Verweis auf eine Dose im Patchfeld, oder aber die IP-Adresse eines weiteren Switches, wenn man sich in einer Umgebung befindet, in der Switches kaskadiert sind. In diesem Fall gibt das *automap* Attribut an, ob das System versuchen soll diesen Switch zu befragen, um den Aufstellungsort des Endgerätes zu ermitteln.

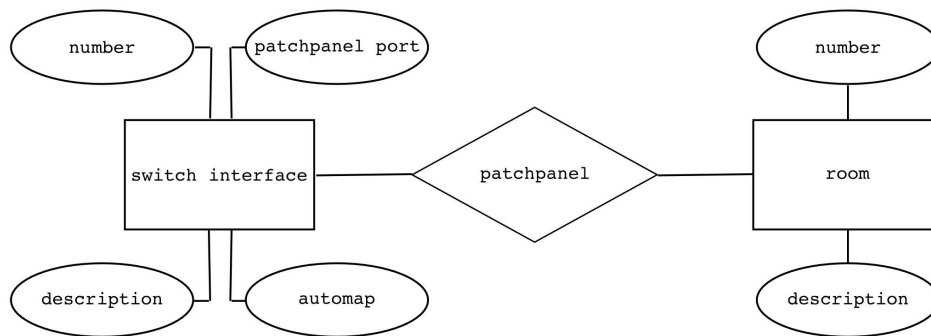


Abbildung B.1: Entity Relation Modell der Datenbank für die Interface - Raum Zuordnung

# Anhang C

## Beispiel eines Erweiterungsskriptes

Das folgende Skript überprüft mit Hilfe des SNMP-Protokolls den Zustand eines Hewlett Packard Color LaserJet 4500; es wird über die in Kapitel 5.7.2 erläuterte Schnittstelle angesprochen. Es werden drei Tests durchgeführt und für jeden dieser Tests wird ein Statuswert zurückgegeben. Der erste Test prüft den Tonerstand der vier im Drucker vorhandenen Kartouchen. Mögliche Rückgabewerte sind:

- &green, wenn in allen Kartouchen mehr als 10% Toner vorhanden ist
- &yellow, wenn in einer der Kartouchen der Tonerstand zwischen 5% und 10% liegt
- &red, wenn in einer der Kartouchen der Tonerstand unterhalb von 5% liegt

Der zweite Test prüft den Zustand der anderen Verbrauchsmaterialien (Transfereinheit, Trommel und Fixierer). Der SNMP-Agent des HP Color LaserJet 4500 unterscheidet hierbei nur zwischen ausreichendem und nicht ausreichendem Zustand der Verbrauchsmaterialien. Eine frühzeitige Warnung ist daher auf diesem Wege nicht möglich. Statuswerte für diesen Test sind daher:

- &green, wenn alle Verbrauchsmaterialien in ausreichender Menge vorhanden sind
- &red, wenn eines der Verbrauchsmaterialien nicht mehr ausreichend für weitere Drucke ist

Der dritte Test prüft die Papierfächer; auch hier kann der SNMP-Agent des Druckers nur zwischen einem leeren Papierfach und einem, das noch Papier enthält, unterscheiden:

- &green, wenn in beiden Fächern noch Papier vorhanden ist
- &yellow, wenn eines der beiden Fächer leer ist
- &red, wenn beide Fächer leer sind

Die zusätzlich zu den Statuswerten zurückgegebenen Details enthalten weitere Angaben darüber, wie das Testergebnis zustande gekommen ist z.B. welche der Tonerkartouchen leer ist und um welches Modell es sich handelt.

Bei dem Erweiterungsskript handelt es sich um ein Shell-Skript. Voraussetzung zum Einsatz dieses Skripts sind die Kommandozeilen-Tools *snmpget* und *snmpwalk*, die Bestandteil von NetSNMP sind.

```
#!/bin/sh

#
# extension skript for HP Color LaserJet 4500
#

HOSTIP=$1
COMMUNITY=$2

# where to find command line snmp tools
SNMPGET="/usr/bin/snmpget"
SNMPWALK="/usr/bin/snmpwalk"

# get general information from printer via snmp
MODEL='$$SNMPGET -v 1 -c $COMMUNITY $HOSTIP host.hrDevice.\
  hrDeviceTable.hrDeviceEntry.hrDeviceDescr.1 2>/dev/null \
  | cut -d " " -f3-'
LOCATION='$$SNMPGET -v 1 -c $COMMUNITY $HOSTIP system.\
  sysLocation.0 | cut -d " " -f3-'
COUNTER='$$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.10.2.1.4.\
  1.1 | cut -d " " -f 4'
MESSAGES='$$SNMPWALK -v 1 -c $COMMUNITY $HOSTIP 43.18.1.1.8 \
  | cut -d " " -f3- | tr -d "\"'

#
# check toner status
#

# get information from printer via snmp
BLACK_UNITNAME='$$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
```

```

1.1.6.1.1 2>/dev/null | cut -d " " -f3- | tr -d "\""
BLACK_STATUS='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.9.1.1 2>/dev/null | cut -d " " -f3- | tr -d "\""

CYAN_UNITNAME='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.6.1.2 2>/dev/null | cut -d " " -f3- | tr -d "\""
CYAN_STATUS='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.9.1.2 2>/dev/null | cut -d " " -f3- | tr -d "\""

MAGENTA_UNITNAME='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.6.1.3 2>/dev/null | cut -d " " -f3- | tr -d "\""
MAGENTA_STATUS='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.9.1.3 2>/dev/null | cut -d " " -f3- | tr -d "\""

YELLOW_UNITNAME='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.6.1.4 2>/dev/null | cut -d " " -f3- | tr -d "\""
YELLOW_STATUS='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.9.1.4 2>/dev/null | cut -d " " -f3- | tr -d "\""

MSG_COUNTER='printf "%-48.48s %6.0f\n" "Page Counter:" \
"$COUNTER"'

# set status message for black toner
if [ "$BLACK_STATUS" = "-3" ]
then
    MSG_BLACK="&green black toner ($BLACK_UNITNAME) ok"
else
    if [ "$BLACK_STATUS" = "-2" ]
    then
        MSG_BLACK="&yellow black toner ($BLACK_UNITNAME) \
is at warning level"
    else
        if [ "$BLACK_STATUS" = "0" ]
        then
            MSG_BLACK="&red black toner ($BLACK_UNITNAME) \
is at panic level"
        fi
    fi
fi

# set status message for cyan toner
if [ "$CYAN_STATUS" = "-3" ]

```

```
then
  MSG_CYAN="&green cyan toner ($CYAN_UNITNAME) ok
"
else
  if [ "$CYAN_STATUS" = "-2" ]
  then
    MSG_CYAN="&yellow cyan toner ($CYAN_UNITNAME) \
    is at warning level"
  else
    if [ "$CYAN_STATUS" = "0" ]
    then
      MSG_CYAN="&red cyan toner ($CYAN_UNITNAME) \
      is at panic level"
    fi
  fi
fi

# set status message for magenta toner
if [ "$MAGENTA_STATUS" = "-3" ]
then
  MSG_MAGENTA="&green magenta toner ($MAGENTA_UNITNAME) ok"
else
  if [ "$MAGENTA_STATUS" = "-2" ]
  then
    MSG_MAGENTA="&yellow magenta toner ($MAGENTA_UNITNAME) \
    is at warning level"
  else
    if [ "$MAGENTA_STATUS" = "0" ]
    then
      MSG_MAGENTA="&red magenta toner ($MAGENTA_UNITNAME) \
      is at panic level"
    fi
  fi
fi

# set status message for yellow toner
if [ "$YELLOW_STATUS" = "-3" ]
then
  MSG_YELLOW="&green yellow toner ($YELLOW_UNITNAME) ok"
else
  if [ "$YELLOW_STATUS" = "-2" ]
  then
```

```
MSG_YELLOW="&yellow yellow toner ($YELLOW_UNITNAME) \  
is at warning level"  
else  
if [ "$YELLOW_STATUS" = "0" ]  
then  
MSG_YELLOW="&red yellow toner ($YELLOW_UNITNAME) \  
is at panic level"  
fi  
fi  
fi  
  
# put all toner messages together  
  
TONER_MSG="$MSG_BLACK  
$MSG_CYAN  
$MSG_MAGENTA  
$MSG_YELLOW  
  
$MSG_COUNTER  
  
Messages:  
$MESSAGES  
"  
# set color for toner status  
  
# default  
TONER_COLOR="green"  
  
if test "$BLACK_STATUS" = "-2" -o "$CYAN_STATUS" = "-2" \  
-o "$MAGENTA_STATUS" = "-2" -o "$YELLOW_STATUS" = "-2"  
then  
TONER_COLOR="yellow"  
fi  
  
if test "$BLACK_STATUS" = "0" -o "$CYAN_STATUS" = "0" \  
-o "$MAGENTA_STATUS" = "0" -o "$YELLOW_STATUS" = "0"  
then  
TONER_COLOR="red"  
fi  
  
#
```

```
# check other consumables
#

# get information from printer via snmp
DRUM_UNITNAME='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.1.\
1.6.1.5 2>/dev/null | cut -d " " -f3- | tr -d "\"'
DRUM_STATUS='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.1.\
1.9.1.5 2>/dev/null | cut -d " " -f3- | tr -d "\"'

TRANSFER_UNITNAME='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.\
11.1.1.6.1.6 2>/dev/null | cut -d " " -f3- | tr -d "\"'
TRANSFER_STATUS='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.\
11.1.1.9.1.6 2>/dev/null | cut -d " " -f3- | tr -d "\"'

FUSER_UNITNAME='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.6.1.7 2>/dev/null | cut -d " " -f3- | tr -d "\"'
FUSER_STATUS='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.11.\
1.1.9.1.7 2>/dev/null | cut -d " " -f3- | tr -d "\"'

# set status message for drum kit
if [ "$DRUM_STATUS" = "-2" ]
then
    MSG_DRUM="&green $DRUM_UNITNAME is ok"
else
    if [ "$DRUM_STATUS" = "0" ]
    then
        MSG_DRUM="&red $DRUM_UNITNAME is at panic level"
    fi
fi

# set status message for transfer kit
if [ "$TRANSFER_STATUS" = "-2" ]
then
    MSG_TRANSFER="&green $TRANSFER_UNITNAME is ok"
else
    if [ "$TRANSFER_STATUS" = "0" ]
    then
        MSG_TRANSFER="&red $TRANSFER_UNITNAME is at panic level"
    fi
fi

# set status message for fuser kit
```

```
if [ "$FUSER_STATUS" = "-2" ]
then
  MSG_FUSER="&green $FUSER_UNITNAME is ok"
else
  if [ "$FUSER_STATUS" = "0" ]
  then
    MSG_FUSER="&red $FUSER_UNITNAME is at panic level"
  fi
fi

# set color for other consumables

# default
CONSUM_COLOR="green"

if test "$DRUM_STATUS" = "0" -o "$TRANSFER_STATUS" = "0" \
-o "$FUSER_STATUS" = "0"
then
  CONSUM_COLOR="red"
fi

# put all consumable messages together
CONSUM_MSG="$MSG_DRUM
$MSG_TRANSFER
$MSG_FUSER

$MSG_COUNTER

Messages:
$MESSAGES
"

#
# check paper
#

PAPER_STATUS_TRAY1='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.\
8.2.1.10.1.1 2>/dev/null | cut -d " " -f3-'
PAPER_STATUS_TRAY2='$SNMPGET -v 1 -c $COMMUNITY $HOSTIP 43.\
8.2.1.10.1.2 2>/dev/null | cut -d " " -f3-'

# default
```

---

```
PAPER_COLOR="green"
PAPER_MSG="&green - both paper trays ok"

if test "$PAPER_STATUS_TRAY1" = "0" \
  -o "$PAPER_STATUS_TRAY2" = "0"
then
  PAPER_COLOR="yellow"
  PAPER_MSG="&yellow - one of the paper trays is emty"
fi

if test "$PAPER_STATUS_TRAY1" = "0" \
  -a "$PAPER_STATUS_TRAY2" = "0"
then
  PAPER_COLOR="red"
  PAPER_MSG="&red - both paper trays are emty"
fi

# put paper messages together
PAPER_MSG="$PAPER_MSG

$MSG_COUNTER

Messages:
$MESSAGES
"

# print all results to stdout
echo "toner
&$TONER_COLOR
'date'
$TONER_MSG
."
echo "consum
&$CONSUM_COLOR
'date'
$CONSUM_MSG
."
echo "paper
&$PAPER_COLOR
'date'
$PAPER_MSG
."
```

# Anhang D

## MIB Ressourcen

Die größte Herausforderung beim Schreiben von Erweiterungsskripten ist die Lokalisierung der gewünschten Information im MIB Baum. Je nach Art und Hersteller des Endgeräts variiert die Anzahl der über SNMP abrufbaren Variablen. Eine Polycom Videokonferenzeinheit stellt über ihren SNMP-Agenten beispielweise außer den Standardvariablen der MIB-2 nur vier Statusvariablen zur Verfügung, die Informationen über den letzten getätigten Anruf erhalten, während ein 3com SuperStack III Switch insgesamt an die 100.000 Variablen vorhält.

Wie in Kapitel 3.5 beschrieben, sind diese Variablen in MIBs definiert und dokumentiert. Die MIBs sind somit unverzichtbar, um die gewünschten Informationen zu finden. Viele Hersteller bieten die MIBs auf ihren Webseiten an; andere wiederum stellen die MIBs auf den Geräten selber zur Verfügung (über http oder ftp zugänglich). Wieder andere stellen ihre MIBs nicht direkt zur Verfügung, sondern erlauben erst auf Anfrage Zugang zu den Informationen. Diverse private Webseiten bieten hierzu umfangreiche Sammlungen von MIB Dateien an. Einige Projekte an Universitäten weltweit erleichtern ebenfalls den Zugang zu den standardisierten MIBs. Die folgenden URLs haben sich auf der Suche nach über SNMP-Agenten zur Verfügung gestellten Informationen als nützlich erwiesen.

- Webbasierter MIB Browser für diverse standardisierte MIBs:
  - <http://www.ibr.cs.tu-bs.de/cgi-bin/sbrowser.cgi>
- hauptsächlich private Seiten mit diversen herstellerspezifischen MIBs:
  - <http://www.wtcs.org/snmp4tpc/mibs.htm>
  - <http://www.mibdepot.com>
  - <http://www.oidview.com/mibs/detail.html>
  - [http://www.somix.com/support/mib\\_resources.php](http://www.somix.com/support/mib_resources.php)

# Anhang E

## Erweiterung der Funktionalität

Bei dem im Rahmen dieser Arbeit entwickelten Netzwerküberwachungssystem handelt es sich um eine Beispielimplementation für dessen Umsetzung nur ein begrenzter Zeitrahmen zur Verfügung stand.

Hauptaugenmerke des hier entwickelten Systems sind einerseits die praktische Umsetzung der zugrunde liegenden Theorie und andererseits die Demonstration der Einsatzmöglichkeiten eines solchen Systems. Während der Konzeptionsphase kamen einige Ideen zur erweiterten Funktionalität des Systems auf, die den praktischen Nutzen erhöhen würden. Einige dieser Ideen sollen hier jedoch zumindest kurz erwähnt werden.

### **Automatische Benachrichtigung**

In vielen Umgebungen ist Verfügbarkeit von größerer Bedeutung als im universitären Umfeld. Gerade wenn diese rund um die Uhr gewährleistet sein soll wäre eine automatische Benachrichtigung der zuständigen Mitarbeiter bei einem (bevorstehenden) Ausfall von Vorteil.

### **Indirekte Informationsgewinnung**

Eine weitere nicht näher angesprochene Informationsquelle über den Zustand von Endgeräten in einem Netzwerk sind solche Geräte, die mit diesen Endgeräten kommunizieren oder zur Kommunikation benutzt werden. Die Bestimmung des Aufstellungsortes eines Endgerätes (vgl. 5.5.14) ist ein Beispiel, aber ein Switch hält wesentlich mehr Informationen über das Netzwerk bereit.

### **Visualisierung der Daten**

Weitere Programmmodule könnten die in der Datenbank abgelegten Daten auswerten und visualisieren, wodurch eine nähere Analyse des gesamten Netzes möglich wäre. Dadurch könnten beispielsweise bei der stetig steigenden Auslastung Prognosen darüber erstellt werden wann die Kapazitäten einzelner Systeme nicht mehr ausreichen.

### **Automatisiertes Management**

Es sind auch Situationen denkbar, in denen das Überwachungssystem selbstständig versuchen könnte diagnostizierte Fehler zu beheben. Auch für einen solchen Zweck bietet SNMP die entsprechende Funktionalität. Beispielsweise könnte das System selbstständig veranlassen, dass temporäre Dateien gelöscht werden, bevor die Plattenkapazität endgültig ausgeschöpft ist.

# Literaturverzeichnis

- [1] Douglas R. Mauro and Kevin J. Schmidt. Essential SNMP. O'Reilly & Associates, Inc., 2001.
- [2] Dirk von Suchodoletz. Ferndiagnose. Linux Magazin, 09:34–43, 2002.
- [3] David Guerrero. Network Management & Monitoring with Linux. Linux Journal, 01, 1997.
- [4] Joe Casad. Teach Yourself TCP/IP in 24 hours. Sams Publishing, 2001.
- [5] Prof. Jürgen Plate. Grundlagen Computernetze. Fachhochschule München, 2000.
- [6] Robin Burk, Thomas Lee, and Martin Bligh. TCP/IP Blueprints. Sams Publishing, 1997.
- [7] William Stallings. SNMP, SNMPv2, SNMPv3 and RMON1 and 2. Addison Wesley, 1999.

# Danksagung

An dieser Stelle möchte ich allen danken, die mich bei der Erstellung dieser Arbeit unterstützt haben. Hervorzuheben sind dabei:

- Dipl. Math. Stefan Koospal für die Ermöglichung dieser Bachelorarbeit, die Bereitstellung der notwendigen Ressourcen und das stete Interesse an meiner Arbeit.
- Frank Schwichtenberg und Tim Oliver Kaiser für die Unterstützung und kreativen Anregungen.
- PD Dr. Hartje Kriete für die Hilfe bei der Einhaltung diverser Formalitäten.
- Prof. Dr. Robert M. Switzer von dem ich den letzten Jahren sehr viel gelernt habe.
- Nicole Duwe für das gewissenhafte Korrekturlesen dieser Arbeit.