Studienarbeit

SNMP in VNUML
Simulationen

Frank Bohdanowicz
Juli 2006

Universität Koblenz-Landau
Abteilung Koblenz

Fachbereich Informatik
AG Rechnernetze und Rechnerarchitektur

Betreuer:
Prof. Dr. Christoph Steigner
Dipl. Inf. Harald Dickel
Hiermit versichere ich, dass ich die vorliegende Studienarbeit selbständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet habe.

Koblenz, den 30.7.2006

______________________________

Frank Bohdanowicz
Zusammenfassung der Studienarbeit


Inhaltsverzeichnis

1 Einleitung .......................................................................................................................... 6
2 Aufbau der Studienarbeit ................................................................................................. 7

3 Grundlagen des Netzwerkmanagements ........................................................................ 8
   3.1 Einleitung .................................................................................................................. 8
   3.2 Die Funktionsbereiche des Netzwerkmanagements ................................................ 9
      3.2.1 Fehlermanagement ......................................................................................... 9
      3.2.2 Konfigurationsmanagement .......................................................................... 10
      3.2.3 Abrechnungsmanagement ............................................................................ 10
      3.2.4 Leistungsmanagement .................................................................................. 11
      3.2.5 Sicherheitsmanagement ................................................................................ 11
   3.3 Zusammenfassung .................................................................................................... 11

4 SNMP-Grundlagen .......................................................................................................... 12
   4.1 Einleitung ................................................................................................................ 12
   4.2 SNMP in den RFCs ................................................................................................. 14
   4.3 Das SNMP Konzept ............................................................................................... 17
      4.3.1 Der Manager .................................................................................................... 18
      4.3.2 Der Agent ........................................................................................................ 19
      4.3.3 Die Managementinformationen ...................................................................... 20
         4.3.3.1 ASN.1 und BER ...................................................................................... 21
         4.3.3.2 Structure of Management Information .................................................... 26
         4.3.3.3 Management Information Base .............................................................. 32
      4.3.4 Das Netzwerkmanagement-Protokoll SNMP .................................................. 33
   4.4 Sicherheit in SNMP .................................................................................................. 39
      4.4.1 SNMPv1 und SNMPv2c ................................................................................ 39
      4.4.2 SNMPv3 .......................................................................................................... 39
         4.4.2.1 Das User-Based Security Model ............................................................... 39
         4.4.2.2 Das View-Based Access Control Model ............................................... 40
   4.5 Erweiterungsmodelle für den SNMP-Agenten ........................................................ 41
      4.5.1 SMUX ............................................................................................................. 41
      4.5.2 AgentX ........................................................................................................... 41
   4.6 Zusammenfassung .................................................................................................... 41

5 Netzwerk-Simulationen mit Virtual Network User Mode Linux ..................................... 42
   5.1 Einleitung ................................................................................................................ 42
   5.2 VNUML-Grundlagen ............................................................................................... 42
   5.3 Durchführung einer VNUML Simulation ................................................................ 45
      5.3.1 Entwurfs Phase ............................................................................................. 46
      5.3.2 Implementierungs Phase ................................................................................ 47
      5.3.3 Ausführungs Phase ......................................................................................... 50
   5.4 Das VNUML Management-Netzwerk ..................................................................... 51
   5.5 User Mode Linux Rechner ..................................................................................... 52
   5.6 Hinweise zur Installation von Software in den UML-Rechner ................................ 60
   5.7 Die Routing-Software Quagga ............................................................................... 63
      5.7.1 Installation ...................................................................................................... 64
      5.7.2 Beispiele für Konfiguration und Umgang mit Quagga ..................................... 66
   5.8 Zusammenfassung .................................................................................................... 69
Anhang

Anleitung:

7 Erstellen eines VNUML-Rechners auf Basis von Gentoo-Linux ............................................. 106
   7.1 Einleitung ......................................................................................................................... 106
   7.2 Erstellen eines UML-Root-Dateisystems auf Basis von Gentoo-Linux ............................. 106
      7.2.1 Vorbereiten des Gentoo-Systems .............................................................................. 107
      7.2.2 Konfiguration des Gentoo-Systems für VNUML ....................................................... 107
      7.2.3 Installation benötigter Software ................................................................................ 114
      7.2.4 Erstellen der Imagedatei ........................................................................................... 117
   7.3 Erstellen eines UML-Kernals ............................................................................................ 119
   7.4 Starten des UML-Rechners .............................................................................................. 120
   7.5 Zusammenfassung ............................................................................................................ 120

Abbildungsverzeichnis ................................................................................................................. 121
Tabellenverzeichnis ...................................................................................................................... 121
Listingverzeichnis ......................................................................................................................... 122
Befehlsverzeichnis ......................................................................................................................... 123
Literaturverzeichnis ....................................................................................................................... 126
1 Einleitung


2 Aufbau der Studienarbeit


In dieser Studienarbeit kommt die folgende Software zum Einsatz:

- Net-SNMP Version 5.2.2
  Net-SNMP Projekt-Homepage: http://www.net-snmp.org

- Quagga Version 0.99.4
  Quagga Projekt-Homepage: http://www.quagga.net

- VNUML Version 1.5
  VNUML Projekt-Homepage: http://jungla.dit.upm.es/~vnuml
3 Grundlagen des Netzwerkmanagements

In diesem Kapitel wird auf die allgemeinen Grundlagen des Netzwerkmanagements eingegangen. Hierzu wird die Definition von Netzwerkmanagement der Internationalen Organisation für Normung (ISO) erläutert, die die Aufgaben des Netzwerkmanagements in fünf Funktionsbereiche untergliedert hat.

3.1 Einleitung


Zu den Aufgaben des Netzwerkmanagements gehört

- das Sammeln von Informationen über die Nutzung des Netzes,
- die Erstellung von Berichten und Statistiken für Planung, Betrieb, Ausfall und Wartung,
- die Konfiguration des Netzes
- sowie die Leistungs-, Ereignis- und Fehlerüberwachung.

Die Erwartungen die in Netzwerkmanagement gesteckt werden, wuchsen entsprechend mit den Einsatzmöglichkeiten von Netzwerken.


Die beiden hervorzuhebenden Standardisierungsorganisationen hierbei sind die Internet Engineering Task Force (IETF) und die Internationale Organisation für Normung (ISO), die unterschiedliche Ansätze bei der Entwicklung von Netzwerkmanagementmodellen verfolgten.

Für die IETF sollte Netzwerkmanagement die folgenden Charakteristiken besitzen:

- möglichst einfach Gehalten
- variablenorientierter Ansatz
- der Austausch von Managementinformationen darf auch unzuverlässig sein (z.B. zugunsten der Stabilität)

Im Gegensatz dazu verfolgte die ISO die folgenden Ansätze:

- möglichst leistungsfähiges Management
- objektorientierter Ansatz
- der Austausch von Managementinformationen muss auf zuverlässige Art und Weise erfolgen

Die ISO realisierte ihre Ansätze der Netzwerkmanagement-Standardisierung im Common Management Information Service (CMIS), was die Dienstprimitive und die Informationsstruktur definiert, sowie dem Common Management Information Protocol (CMIP) für den Nachrichtenaustausch. Die ISO entwickelte CMIP/CMIS jedoch als Grundlage für ihr OSI-Management, für die Verwaltung von Netzwerken die auf dem Open Interconnection System Reference Model (OSI) basieren, das die bestehenden TCP/IP Netze auch bearbeiten sollte. Da jedoch die breite Einführung OSI basierter Netze auf sich warten ließ, wurde schließlich mit CMOT (CMIP over TCP/IP) das OSI-Protokoll für das Netzwerkmanagement auf einem TCP/IP-Protokollstack entwickelt, das sich auf der Anwendungsschicht wie CMIP verhält.


3.2 Die Funktionsbereiche des Netzwerkmanagements

Im Zusammenhang mit der Entwicklung des OSI-Managements führte die ISO das Konzept der spezifischen Management-Funktionsbereiche unter der Bezeichnung SMFA's (System Management Functional Area) ein und definierte fünf funktionelle Bereiche, die die Verwaltung eines Netzwerks beschreiben und die auch unter dem Begriff FCAPS bekannt sind. FCAPS ist das Akronym für Faults, Configuration, Accounting, Performance und Security, in deutscher Übersetzung also, Fehler, Konfiguration, Abrechnung, Leistung und Sicherheit. Im folgenden werden diese fünf Funktionsbereiche des Netzwerkmanagements näher erläutert.

3.2.1 Fehlermanagement

Unter dem Begriff Fehlermanagement werden alle Funktionen des Netzwerkmanagements zusammengefasst, die zur Fehlerprophylaxe, Fehlererkennung und Fehlerbehebung im Netzwerk benötigt werden. Der Begriff Fehler steht dabei für unerwünscht auftretende Ereignisse, die sich negativ auf die Funktion und den Betrieb des Netzwerks auswirken. Das vermeiden, identifizieren, isolieren, protokollieren und korrigieren dieser Ereignisse ist die Hauptaufgabe des Fehlermanagements.

Fehlermanagement ist ein sehr wichtiger Bereich des Netzwerkmanagements, denn es dient dazu das Netzwerk in einem verfügbaren Zustand zu halten, bzw. diesen wiederherzustellen. Die Fähigkeit einen Fehler zu erkennen und zu beheben hat Vorrang vor allen anderen Bereichen des Netzwerkmanagements, denn ein Netzwerk muss generell fehlerfrei funktionieren um seine Aufgaben überhaupt erfüllen zu können.

Die Fehlererkennung in Netzwerken kann in zwei Bereiche untergliedert werden.

- Reaktive Fehlererkennung
  Hierbei ist der Fehler bereits eingetreten, die beteiligten Netzwerk-Komponenten sind aber dazu fähig auf den Fehler zu reagieren und darauf aufmerksam zu machen.
Eine Reaktive Fehlererkennung wäre zum Beispiel, wenn ein Router feststellt, dass ein an ihn angeschlossenes Gerät nicht mehr antwortet und er darauf reagiert, in dem er eine Fehlermeldung erzeugt, in der der Fehler beschrieben ist.

- **Proaktive Fehlererkennung**
  Die proaktive Fehlererkennung befasst sich mit der frühzeitigen Erkennung von Problemen, die Fehler zur Folge haben können. Eine solche Fehlererkennung hätte zum Beispiel ein Datei-Server der feststellt, dass seine Festplatte zu 90% mit Daten gefüllt ist und der daraufhin eine Fehlermeldung erzeugt, in der er vor der zu vollen Festplatte warnt.


### 3.2.2 Konfigurationsmanagement


Je nach Netzgröße und Systemvielfalt werden unterschiedliche Anforderungen an das Konfigurationsmanagement gestellt. Dieser Managementbereich bietet heute wohl die meisten Tools, auch weil jeder Hersteller netzwerkfähiger Komponenten herstellereigene Standards aufbaut, um seinen Produkten eine komfortable Zugangsschnittstelle für die Konfiguration zu bieten und auch um eher homogene Netzwerke mit Produkten aus dem eigenen Angebot zu fördern.

### 3.2.3 Abrechnungsmanagement

Das Abrechnungsmanagement befasst sich mit Funktionen zur ordnungsgemäßen Abwicklung der Benutzung des Netzwerks, wie zum Beispiel Zugangsverwaltung, Verbraucherkontrolle, Abrechnungshilfe und angebotene Mehrwertdienste.


### 3.2.4 Leistungsmanagement


3.2.5 Sicherheitsmanagement

Das Sicherheitsmanagement beinhaltet alles was in einem Netzwerk mit Sicherheit und dem Schutz von Informationen zu tun hat. Es umfasst aber auch den Schutz von Objekten, von Diensten und Ressourcen. Sicherheitsmanagement dient der Identifizierung von Risiken aller Art, die den reibungslosen Betrieb des Netzwerks beeinträchtigen können. Das Sicherheitsmanagement hat dafür Sorge zu tragen, dass das Netzwerk gegenüber diesen Risiken abgesichert ist.

Sicherheitsmanagement kann in zwei Ebenen untergliedert werden.

- **Physikalische Sicherheit**
  Diese Ebene der Sicherheit befasst sich mit der physikalischen und lokalen Isolation von Netzwerkkomponenten zum Schutz vor äußeren, physikalischen Einflüssen.

- **Logische Sicherheit**

Sicherheitsmanagement umfasst auch sicherheitspolitische Aspekte und muss ethische und gesetzliche Komponenten, wie zum Beispiel gesetzliche Datenschutzbestimmungen, ebenso berücksichtigen wie wirtschaftliche Voraussetzungen. Das Sicherheitsmanagement muss unter der Prämisse geplant werden, dass Informationen einen Wert darstellen, der quantifiziert und qualifiziert werden kann und mit entsprechendem Aufwand geschützt werden muss.

3.3 Zusammenfassung


4 SNMP-Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen von SNMP erläutert. Aufbau und Funktionsweise des Protokolls erlauben ein besseres Nachvollziehen der Einsatzmöglichkeiten im Netzwerkmanagement.

4.1 Einleitung


Mit SNMP sind hauptsächlich die folgenden Managementaufgaben erfüllbar:

- Überwachung von Netzwerk-Komponenten
- Fernsteuerung und Fernkonfiguration von Netzwerk-Komponenten
- Fehlererkennung und Fehlerbenachrichtigung


Darüber hinaus bietet SNMP dem Agenten die Möglichkeit von sich aus seinem Manager eine Ereignismeldung zu zusenden, die diesen über ein im Zuständigkeitsbereich des Agenten eingetretenen Ereignissen unterrichtet.

Damit Agenten und Manager, auch verschiedener Hersteller, die gleiche Managementinformation auch auf gleiche Weise behandeln besitzt jeder Agent eine Beschreibung seiner Managementinformationen in einer Datendefinitionssprache, genannt „Structure of Management Information“. Mit Hilfe dieser Definitionssprache werden die Managementinformationen zur Identifizierung in eine logische Struktur abgebildet, die „Management Information Base“ genannt wird.


In der Praxis hat sich SNMP in erster Linie als Frage/Antwortprotokoll durchgesetzt das hauptsächlich Aufgaben des Leistungsmanagements wahrnimmt und Zustands-Informationen sowie Leistungsdaten von verschiedenen, vor allem ressourcennarmen, Netzwerk-Geräten an einer zentralen Stelle im Netzwerk sammelt.


Abb. 4.2: Historische Entwicklung von SNMP
4.2 SNMP in den RFCs


Einige Stadien, die RFCs innehaben können, sind:

- **Informational** - zur Information, Hinweis, Idee
- **Proposed Standard** - Vorschlag zum Standard
- **Draft Standard** - Entwurf zum Standard, Begutachtung von mindestens zwei voneinander unabhängigen Implementierungen ist nun Voraussetzung für Standard
- **(Full) Standard** - offizieller Standard und zur allgemeinen Anwendung empfohlen
- **Experimental** - zum Experimentieren
- **Historic** - ausgemustert, in der Praxis nicht mehr zur Anwendung empfohlen

![Diagramm des Standardisierungsprozesses der IETF](image-url)


Bezüglich der von der IETF definierten Standards kann SNMP in drei Komponenten untergliedert werden:


- einer standardisierten „Sammelstelle“ für die in SMI definierten Managementinformationen. Auf Basis von SMI ist die „Management Information Base“ (MIB) definiert, eine Art Katalog oder auch textuelle Datenbank, die alle Definitionen der Managementinformationen eines Agenten beinhaltet und diese logisch strukturiert. Eine standardisierte Ausgabe der MIB, mit Definitionen von Managementinformationen bzgl. allgemeiner und herstellerübergreifender, technischer Bereiche des Internets, bietet eine Grundlage dessen, was eine Managementstation an Informationen versteht sollte und was ein Agent an Managementinformationen, bzgl. der technischen Bereiche, die er unterstützt, liefern sollte. Diese MIB liegt derzeit in der Version zwei vor (MIB-II). Es gibt aber zahlreiche Aktualisierungen mit neuen Definitionen von Managementinformationen die allgemeine technische Bereiche des Internets betreffen und als Update des Standards über weitere RFC-Dokumente veröffentlicht wurden. Private Organisationen können zusätzlich eigene MIBs für die Managementinformationen ihrer Produkte anbieten, die jedoch selten als RFC-Dokument vorliegen und auch nicht von der IETF standardisiert werden.


Überblick und Einleitung zum gegenwärtigen Internet-Standard Management Framework, SNMPv3, sind im RFC 3410 zu finden.
In der folgenden Tabelle sind die wichtigsten RFCs bzgl. des SNMP-Management Frameworks aufgelistet.

<table>
<thead>
<tr>
<th>RFC-ID</th>
<th>Titel</th>
<th>gegenwärtiger Status</th>
<th>Datum der Veröffentlichung</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP Version 1 (SNMPv1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1155 (STD 16)</td>
<td>Structure and identification of management information for TCP/IP-based internets</td>
<td>STANDARD</td>
<td>Mai 1990</td>
</tr>
<tr>
<td>1156</td>
<td>Management Information Base for network management of TCP/IP-based internets</td>
<td>HISTORIC</td>
<td>Mai 1990</td>
</tr>
<tr>
<td>1157 (STD 15)</td>
<td>Simple Network Management Protocol (SNMP)</td>
<td>HISTORIC</td>
<td>Mai 1990</td>
</tr>
<tr>
<td>1212 (STD 16)</td>
<td>Concise MIB definitions</td>
<td>STANDARD</td>
<td>März 1991</td>
</tr>
</tbody>
</table>

Community-Based SNMP Version 2 (SNMPv2c)

<table>
<thead>
<tr>
<th>RFC-ID</th>
<th>Titel</th>
<th>gegenwärtiger Status</th>
<th>Datum der Veröffentlichung</th>
</tr>
</thead>
<tbody>
<tr>
<td>1901</td>
<td>Introduction to Community-based SNMPv2</td>
<td>HISTORIC</td>
<td>Januar 1996</td>
</tr>
</tbody>
</table>

Structure of Management Information Version 2 (SMIv2)

<table>
<thead>
<tr>
<th>RFC-ID</th>
<th>Titel</th>
<th>gegenwärtiger Status</th>
<th>Datum der Veröffentlichung</th>
</tr>
</thead>
<tbody>
<tr>
<td>2578 (STD 58)</td>
<td>Structure of Management Information Version 2 (SMIv2)</td>
<td>STANDARD</td>
<td>April 1999</td>
</tr>
<tr>
<td>2579 (STD 58)</td>
<td>Textual Conventions for SMIv2</td>
<td>STANDARD</td>
<td>April 1999</td>
</tr>
<tr>
<td>2580 (STD 58)</td>
<td>Conformance Statements for SMIv2</td>
<td>STANDARD</td>
<td>April 1999</td>
</tr>
</tbody>
</table>

SNMP Version 3 (SNMPv3)

<table>
<thead>
<tr>
<th>RFC-ID</th>
<th>Titel</th>
<th>gegenwärtiger Status</th>
<th>Datum der Veröffentlichung</th>
</tr>
</thead>
<tbody>
<tr>
<td>3410</td>
<td>Introduction and Applicability Statements for Internet-Standard Management Framework</td>
<td>INFORMATIO-NAL</td>
<td>Dezember 2002</td>
</tr>
<tr>
<td>3415 (STD 62)</td>
<td>View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)</td>
<td>STANDARD</td>
<td>Dezember 2002</td>
</tr>
</tbody>
</table>

Tabelle 4.1: Einige wichtige RFC-Dokumente zu SNMP (Quelle: www.rfc-editor.org, Stand: Juli 2006)
4.3 Das SNMP Konzept


Mit SNMP wird für gewöhnlich nicht nur das eigentliche Netzwerkmanagement-Protokoll bezeichnet, sondern auch die zugehörige Rahmenarchitektur, das Standard-Internet Management Framework.

Das SNMP-Modell eines verwalteten Netzes besteht aus vier Komponenten:

- den Managementstationen, genannt Manager
- den verwalteten Netzwerk-Komponenten, genannt Agenten
- den Managementinformationen (Objekt-Ressourcen)
- dem Netzwerkmanagement-Protokoll SNMP


Die Agenten wiederum können selbständig dem Manager Ereignismeldungen zusenden, wenn sie bestimmte Ereignisse, in dem von ihnen verwalteten System, registrieren.


Die folgende Abbildung gibt einen Überblick über die Netzwerkmanagement-Architektur des SNMP Modells.

![SNMP-Modell](image-url)

### 4.3.1 Der Manager


Die Management-Applikationen des Managers interagieren mit den Agenten über das Netzwerkmanagement-Protokoll SNMP. Das SNMP-Modell erlaubt eine hierarchische Struktur, was bedeutet, dass Manager selbst wiederum Agenten anderer Manager sein können. Für die volle Funktionalität der Management-Applikationen benötigt der Manager die textuelle Abbildung der „Management Information Base“ (MIB), in der die Objekte aller Managementinformationen, die die angeschlossenen Agenten bieten, über die Datendefinitionssprache SMI beschrieben sind. Für die Funktion des SNMP-Managers selbst ist das Vorhandensein einer MIB jedoch keine Voraussetzung, da die Managementinformationen letztlich über eine Identifikationsnummer vom Agenten abgerufen werden, die aber aus der MIB ausgelesen werden kann. Mit sogenannten MIB-Browisern verfügt der Manager über eine Management-Applikation, die ihm die zur Verfügung stehenden Managementinformationen aufzeigt. Die MIB des Managers dient somit in erster Linie dem Menschen zum Verstehen der verwalteten Managementinformationen.


![Abb. 4.5: SNMP-Architektur in TCP/IP-Netzen](image-url)
Das MPS übernimmt die Nachricht vom Dispatcher, verarbeitet sie entsprechend der angegebenen SNMP Version und reicht sie dann an das Security Subsystem weiter. Im Security Subsystem werden die Nachrichten entsprechend dem verwendeten Sicherheitsmodell ver- und entschlüsselt.

Das folgende Schema zeigt den modularen Aufbau des SNMPv3-Managers:

![Abb. 4.6: Architektur des SNMPv3-Managers](image)

### 4.3.2 Der Agent


Für das SNMP-Modell existieren auch verschiedene Erweiterungsmodelle für den Agenten über die ein SNMP-Agent in eine verteilte Master- und Sub-Agenten Struktur aufgeteilt werden kann. Der Master-Agent dient dann als Schnittstelle zwischen seinen verschiedenen Sub-Agenten und dem Manager. Die Sub-Agenten sind zum Beispiel Sub-Prozesse unabhängiger, eigenständiger Programme, die über keinen eigenen SNMP-Agenten verfügen. Diese kommunizieren nicht über das Netzwerkmanagement-Protokoll SNMP mit ihrem Master-Agenten, sondern über spezielle Protokolle, die Erweiterungsmodelle wie SMUX (SNMP Multiplexing) oder AgentX (Agent eXtensibility) bereitstellen.


Mit diesem zusätzlichen Subsystem wird der Zugriff auf die verwalteten Objekte geregelt.
Das nachfolgende Schema zeigt den modularen Aufbau des SNMPv3-Agenten:

4.3.3 Die Managementinformationen


Um Managementinformationen mit Hilfe von Objekten definieren zu können, bedarf es plattformübergreifender und herstellerunabhängiger Standards, die im folgenden beschrieben werden.
4.3.3.1 ASN.1 und BER


Tabelle 4.2 zeigt die lexikalischen Konventionen, die in SMI gelten:

<table>
<thead>
<tr>
<th>Art des Labels</th>
<th>lexikalische Konvention</th>
<th>Beispiel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrierte Basis-Datentypen</td>
<td>Alles in Großbuchstaben</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Benutzerdefinierte Datentypen</td>
<td>Das Initial ist großgeschrieben</td>
<td>DisplayString</td>
</tr>
<tr>
<td>Bezeichner</td>
<td>Das Initial ist kleingeschrieben</td>
<td>sysDescr</td>
</tr>
<tr>
<td>Leerräume (z.B. Tabulatoren, Zeilenschaltungen, usw) haben keinerlei Bedeutung</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kommentare beginnen mit -- und enden beim nächsten Vorkommen von --</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Basis-Datentyp</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>Ganzzahl mit arbiträrer Länge</td>
</tr>
<tr>
<td>BIT STRING</td>
<td>Zeichenkette mit 0 oder mehr Bit</td>
</tr>
<tr>
<td>OCTET STRING</td>
<td>Zeichenkette mit 0 oder mehr vorzeichenlosen Byte</td>
</tr>
<tr>
<td>NULL</td>
<td>Platzhalter</td>
</tr>
<tr>
<td>OBJECT IDENTIFIER</td>
<td>Objektbezeichner (offiziell definierter Datentyp)</td>
</tr>
</tbody>
</table>

Tabelle 4.3: Basis-Datentypen, die SNMP verwenden kann

Der INTEGER Typ kann theoretisch jeden ganzzahligen Wert annehmen. Im Beispiel wird die INTEGER Variable „count“ auf 100 gesetzt.

```asciidoc
count INTEGER := 100
```

*Listing 4.1: Definition eines Integers in ASN.1*

Meist werden auf der Basis von INTEGER neue Sub-Datentypen definiert, deren Variablen auf bestimmte Werte oder einen bestimmten Bereich eingeschränkt sind. Im Beispiel wird ein Wert „Status“ definiert der den Wert 1 für „up“, 2 für „down“ oder 3 für „unknown“ einnehmen kann und ein Wert „PacketSize“, dessen Zahlenwert zwischen 0 und 1023 liegt.

```asciidoc
Status ::= INTEGER { up(1), down(2), unknown(3) }
PacketSize ::= INTEGER (0..1023)
```

*Listing 4.2: Definition eines wertebeschränkten Integer-Datentypen in ASN.1*

Objekte vom Typ BIT STRING oder OCTET STRING enthalten Null oder mehr Bits, bzw. Bytes, wobei ein Bit entweder 0 oder 1 ist und ein Byte in den Bereich von 0 bis 255 fällt.

Die OID eines Objektes ergibt sich aus den Beschrifungen oder Nummern der Knoten, die auf dem Weg von der Wurzel des Registrierungshauses bis zum Objekt selbst liegen, jeweils getrennt durch einen Punkt. Die Objekte der Gruppe „system“, die eine Beschreibung des SNMP-Agenten beinhalten und über die jeder SNMP-Agent verfügen sollte, sind somit alle unter der OID „1.3.6.1.2.1.1...“ zu erreichen. Das Objekt „sysDescr“ der Gruppe „system“ ist unter der OID „1.3.6.1.2.1.1.1“ oder unter „iso.org.internet.mgmt.mib-2.system.sysDescr“ zu finden. Auch Mischformen der beiden Notationen sind erlaubt. Um die Managementinformation selbst zu erhalten muss an die OID des Objektes noch ein Instanzwert angehängt werden.

Um aus den in Tabelle 4.3 aufgezählten primitiven Basis-Datentypen neue, komplexere zu bilden werden in SNMP die aus ASN.1 bekannten Konstruktionen SEQUENCE und SEQUENCE OF verwendet. SEQUENCE bildet ein geordnete Typenliste und SEQUENCE OF ein eindimensionales Array bestehend aus einem einzelnen Typ. Analog gibt es in noch SET und SET OF für ungeordnete Listen sowie CHOICE, das aus einer bestimmten Typenliste ein Union erstellt.


ASN.1 definiert auch einen komplexen aber mächtigen Makromechanismus, der häufig in SNMP Verwendung findet. Ein Makro kann als eine Art Prototyp benutzt werden, um ein Set neuer Typen und Werte mit eigener Syntax zu erstellen. Jedes Makro definiert einige Schlüsselwörter, die im Aufruf benutzt werden, um die einzelnen Parameter zu definieren. Im Kapitel 4.3.3.2 werden einige Makros aus dem SNMP Umfeld vorgestellt.


Mit BER kodierte Daten bestehen immer aus einer Bytekette, enthalten also immer ein durch acht teilbare Anzahl von Bits. Jeder Übertragene Wert besteht dabei aus vier Teilen.

1. Bezeichner ID (Typ oder Tag)
2. Länge der Inhaltsdaten in Byte
3. Inhaltsdaten
4. Inhaltsende-Flag, nur falls die Länge der Inhaltsdaten unbekannt ist (End-of-content)


Der erste Teil identifiziert den verwendeten Datentyp. Die ersten zwei High-Order-Bits identifizieren die Klasse des verwendeten Datentyps. Die verwendeten Bits sind hier 00 für universelle, 01 für anwendungsweite, 10 für kontextspezifische und 11 für private Datentypen. Das dritte Bit gibt an, ob es sich um einen einfachen (0) oder um einen zusammengesetzten Datentyp (1) handelt. Die letzten fünf Bits kodieren die Tag-Nummer, falls diese im Bereich von 0 bis 30 liegt. Liegt sie darüber enthalten diese fünf Low-Order-Bits 11111, mit dem wahren Wert der Tag-Nummer in den nächsten Bytes.

<table>
<thead>
<tr>
<th>Code-Nummer</th>
<th>Datentyp</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIVERSAL 2</td>
<td>INTEGER</td>
</tr>
<tr>
<td>UNIVERSAL 3</td>
<td>BIT STRING</td>
</tr>
<tr>
<td>UNIVERSAL 4</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>UNIVERSAL 5</td>
<td>NULL</td>
</tr>
<tr>
<td>UNIVERSAL 6</td>
<td>OBJECT IDENTIFIER</td>
</tr>
<tr>
<td>UNIVERSAL 16</td>
<td>SEQUENCE und SEQUENCE-OF</td>
</tr>
</tbody>
</table>

Tabelle 4.4: BER-Kodierungsnummern einiger universeller Datentypen

Nach dem Bezeichner folgt die Angabe der Länge der Inhaltsdaten. Längenangaben von von weniger als 128 Byte können direkt im ersten Bytefeld kodiert werden, wobei das High-Order-Bit auf 0 gesetzt ist. Übersteigt die Anzahl der Inhaltsdaten die Länge von 128 Byte, so genügt ein Bytefeld nicht zur Angabe der Länge und es werden mehrere Bytefelder benötigt. In diesem Fall wird das erste Bytefeld dazu verwendet, die Anzahl der für die Längenangabe verwendeten Bytefelder anzugeben. Dabei ist das High-Order-Bit im ersten Bytefeld gesetzt und zeigt mit seinen restlichen 7 Bitwerten an, wie viele der noch folgenden Bytefelder allein für die Längenangabe verwendet werden. Benötigen die Inhaltsdaten zum Beispiel 1000 Byte, so steht im ersten Bytefeld der Längenangabe die Zahl 130 (10000010), um anzuzeigen, dass ein 2-Byte-Längenfeld (0000011111010000) mit der Längenangabe 1000 folgt.

Die Kodierung der Inhaltsdaten hängt vom verwendeten Datentyp ab.
Ganzzahlen werden im Zweier-Komplement kodiert. Eine positive ganze Zahl unter 128 erfordert ein Byte, eine unter 32.768 zwei Byte, usw. .

Bitfolgen werden auch als solche kodiert. Da die Länge der Inhaltsdaten in Bytes und nicht in Bit angegeben ist, wird vor der eigentlichen Übertragung der Bitfolge ein zusätzliches Bytefeld übertragen, in dem angegeben wird, wie viele Bits im letzten Bytefeld, das die Bitfolge belegt, unbenutzt sind.

Die Bytes der Octetfolgen werden einfach standardmäßig von links nach rechts übertragen.

Der Nullwert wird angegeben in dem das Längenfeld auf 0 gesetzt wird.

Ein Object Identifier wird als Folge der darstellenden Ganzzahlen kodiert. Der Object Identifier „internet“ ist zum Beispiel {1, 3, 6, 1}. Da die erste Zahl allerdings immer 0, 1 oder 2 und die zweite nach Definition weniger als 40 ist, werden die ersten zwei Zahlen a und b, als 1 Byte mit dem Wert 40 * a + b kodiert. Beim Object Identifier „internet“ ist diese Zahl 43. Zahlen die größer als 127 sind werden, wie bei der Längenangabe der Inhaltsdaten, über mehrere Bytes kodiert, wobei im ersten Byte das High-Order-Bit gesetzt ist und mit den übrigen sieben Bits die Anzahl der für die Längenangabe verwendeten Bytefelder angegeben wird.

Im folgenden Beispiel wird ein komplexerer Datensatz in ASN.1 strukturiert und anschließend mittels den BER Kodierungsregeln codiert. Jede Maschine, die über diese ASN.1 Datentypdefinition verfügt, kann den Datensatz vollständig rekonsitrieren.

<table>
<thead>
<tr>
<th>Datendefinition:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host:</td>
</tr>
<tr>
<td>Hostname: Server13</td>
</tr>
<tr>
<td>IP-Address: 192.168.1.13</td>
</tr>
</tbody>
</table>

ASN.1 Datentypdefinition:

Host ::= [APPLICATION 10] SEQUENCE {
  Hostname HostID,
  IP-Address IPAddress
}

HostID ::= [APPLICATION 11] SEQUENCE {
  Name OCTET STRING
  Number INTEGER
}

IPAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING( SIZE( 4 ) )

BER-Kodierung:

<table>
<thead>
<tr>
<th>Host</th>
<th>Länge</th>
<th>Inhalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>6a</td>
<td>16</td>
<td>SEQ.</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
<td>HostID</td>
</tr>
<tr>
<td>0d</td>
<td></td>
<td>6b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Länge</th>
<th>Inhalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HostID</th>
<th>Länge</th>
<th>Inhalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>6b</td>
<td>0d</td>
<td>Name</td>
</tr>
<tr>
<td>04</td>
<td>06</td>
<td>04</td>
</tr>
<tr>
<td>30</td>
<td>0b</td>
<td>Number</td>
</tr>
<tr>
<td>04</td>
<td>01</td>
<td>0d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IPAddress</th>
<th>Länge</th>
<th>Inhalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td></td>
<td>04</td>
</tr>
</tbody>
</table>

vollständige Hex-Darstellung der BER-Kodierung:

6a 16 30 14 6b 0d 30 0b 04 06 53 65 72 76 65 72 02 01 0d 04 c0 a8 01 0d

Abb. 4.11: Beispiel für die Kodierung eines komplexeren ASN.1 Datensatzes mit BER

4.3.3.2 Structure of Management Information

Structure of Management Information (SMI) ist die Struktur, in der die Managementinformationen für den Umgang mit SNMP definiert sein müssen. Es handelt sich dabei um eine Norm, die Regeln dafür aufstellt, wie die Managementinformationen, unter Verwendung von ASN.1, in Objekten organisiert, typisiert und beschrieben werden. Der SMI Ansatz ist zwar etwas bürokratisch aber auch wichtig, damit hunderte von Produkten verschiedenster Hersteller über eine Managementplattform kommunizieren können. SMI liegt derzeit in der Version zwei vor, genannt SMIv2, und wurde von der IETF über die RFCs des STD 58 standardisiert. Die vorherige Version SMIv1 ist nach wie vor ein Standard (STD 16), wird allerdings nicht mehr zur Anwendung empfohlen.

In der folgenden Tabelle sind in SMIv2 gültigen Datentypen aufgelistet.

<table>
<thead>
<tr>
<th>Bezeichnung</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer32</td>
<td>Numerisch</td>
<td>Ganzzahl zwischen -2147483648 und 2147483647 (-2^{31} und 2^{31}-1), auch in 64 Bit Systemen</td>
</tr>
<tr>
<td>Unsigned32</td>
<td>Numerisch</td>
<td>Ähnlich Integer32, jedoch vorzeichenlos</td>
</tr>
<tr>
<td>Gauge32</td>
<td>Numerisch</td>
<td>Pegel, vorzeichenlose 32-Bit Zahl, kann steigen oder fallen, behält aber den erreichten Höchstwert bei und beginnt nicht automatisch von vorn, Zum Beispiel die Länge der Ausgabe-Warteschlange an einer Schnittstelle.</td>
</tr>
<tr>
<td>Counter32 / Counter64</td>
<td>Numerisch</td>
<td>Zähler, steigt bis zu einem Maximalwert an und beginnt dann automatisch wieder bei Null. Zum Beispiel die Anzahl der an einem Port empfangenen Datenpakete</td>
</tr>
<tr>
<td>TimeTicks</td>
<td>Numerisch</td>
<td>Zeiteinheit, eine nicht negative ganze 32-Bit Zahl, welche die Zeit in hundertstel Sekunden zählt. Zum Beispiel die Berechnung der Systembetriebszeit</td>
</tr>
<tr>
<td>IpAddress</td>
<td>Zeichenkette</td>
<td>Eine 32 Bit-IPv4-Adresse.</td>
</tr>
<tr>
<td>Opaque</td>
<td>Zeichenkette</td>
<td>Datentyp für beliebige Kodierung. Erlaubt das Einpacken jedes beliebigen ASN.1 Typs in einen OCTET STRING</td>
</tr>
<tr>
<td>BITS</td>
<td>Zeichenkette</td>
<td>Eine Aufzählung von benannten Bits.</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Liste, die null oder mehr Objekte enthält</td>
<td></td>
</tr>
<tr>
<td>SEQUENCE OF</td>
<td>Array, eine Liste von Listen</td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 4.5: Einige SMIv2 Datentypen für SNMP Management-Objekte
Zusätzlich definiert SMIv2 über die RFCs des STD 58 noch einige Textkonventionen, die eine abstraktere Definition von Objekten erlauben und auf den Basis-Datentypen aufbauen. Einige dieser Textkonventionen sind in der folgenden Tabelle aufgelistet:

<table>
<thead>
<tr>
<th>Name</th>
<th>SMI Basistyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayString</td>
<td>OCTET STRING</td>
<td>Ein aus NVT ASCII-Zeichen bestehende Zeichenkette von maximal 255 Zeichen Länge</td>
</tr>
<tr>
<td>PhysAddress</td>
<td>OCTET STRING</td>
<td>Eine physikalische Adresse</td>
</tr>
<tr>
<td>MacAddress</td>
<td>OCTET STRING</td>
<td>Eine durch OCTET STRING der Länge 6 dargestellte MAC-Adresse nach dem IEEE 802 Standard (Hardware Adresse einer Ethernet Karte)</td>
</tr>
<tr>
<td>TruthValue</td>
<td>INTEGER</td>
<td>Über zwei Integer Werte werden Boolean Werte definiert. (1 = true, 2 = false)</td>
</tr>
<tr>
<td>TestAndIncr</td>
<td>OBJECT IDENTIFIER</td>
<td>Ermöglicht atomare Operationen und verhindert, dass mehrere Manager auf ein Objekt gleichzeitig zugreifen (im Sinne einer Semaphor).</td>
</tr>
<tr>
<td>Autonomie-Type</td>
<td>OBJECT IDENTIFIER</td>
<td>Stellt einen unabhängigen, erweiterbaren Typ-Identifikations-Wert dar. Ermöglicht einen Object Identifier zur Definition eines Unterbaumes mit zusätzlichen Definitionen.</td>
</tr>
<tr>
<td>VariablePointer</td>
<td>OBJECT IDENTIFIER</td>
<td>Ein Zeiger auf eine bestimmte Instanz eines Objektes</td>
</tr>
<tr>
<td>RowPointer</td>
<td>OBJECT IDENTIFIER</td>
<td>Ein Zeiger auf eine Tabellenzeile</td>
</tr>
<tr>
<td>RowStatus</td>
<td>INTEGER</td>
<td>Überwacht den Status und die Konsistenz einer Tabelle, insbesondere, wenn mehrere Manager darauf zugreifen und die Spalten einer Tabelle verändern. Besitzt die folgenden Status angaben: active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>TimeTicks</td>
<td>Der sysUpTime Objekt Wert, zum Zeitpunkt eines bestimmten Ereignisses. Misst also die Zeit vom Start des Systems bis zum Eintritt des Ereignisses.</td>
</tr>
<tr>
<td>TimeInterval</td>
<td>INTEGER</td>
<td>Eine Zeitperiode, gemessen in Einheiten von 0,01 Sekunden</td>
</tr>
<tr>
<td>DateAndTime</td>
<td>OCTET STRING</td>
<td>Beschreibt Datum und Zeit</td>
</tr>
<tr>
<td>StorageType</td>
<td>INTEGER</td>
<td>Beschreibt die Art von Speicher, die ein Agent benutzt. Mögliche Werte sind [RFC 2579]: other(1) - eh? volatile(2) - Daten gehen bei Reboot verloren (bsp.: RAM) nonVolatile(3) - Daten bleiben permanent erhalten (bsp.: NVRAM) permanent(4) - Daten können verändert aber nicht gelöscht werden (bsp.: z.T. in ROM) readOnly(5) - Daten können weder verändert noch gelöscht werden (bsp.: ROM)</td>
</tr>
<tr>
<td>TDomain</td>
<td>OBJECT IDENTIFIER</td>
<td>Beschreibt eine Art von Transport Service</td>
</tr>
<tr>
<td>TAddress</td>
<td>OCTET STRING</td>
<td>Beschreibt eine Transport Service Adresse</td>
</tr>
</tbody>
</table>

Tabelle 4.6: Einige SMIv2 Textkonventionen (Entnommen RFC 2579)

Darüber hinaus können über ein Makro auch eigene Textkonventionen, die auf vorhandenen Datentypen oder anderen Textkonventionen aufbauen, definiert werden. Im Beispiel rechts wird die Textkonvention „RunState“, die vier verschiedene Zustände einnehmen kann, auf Basis des Datentyps INTEGER definiert. Die Parameter des Makros sind weitestgehend selbsterklärend, werden aber im Zusammenhang mit einem weiteren wichtigen Makro des SNMP-Modells im folgenden noch erläutert.

```
RunState ::= Textual Convention
  STATUS current
  DESCRIPTION "…"
  SYNTAX INTEGER {
    running(1)
    runable(2)
    waiting(3)
    exiting(4) }
```

Listing 4.4: Definition der Textkonvention „RunState“


Alle Objekt-Definitionen werden in ASCII Text-Code innerhalb einer oder mehrerer Textdateien notiert, die hier schlicht MIB-Dateien genannt werden. Das Gerät, das alle Objekt-Definitionen innerhalb der MIB-Dateien umgibt, ist die „Management Information Base“ (MIB).

### Listing 4.5: Textuelles Gerüst der MIB-Dateien


### Listing 4.6: Definition eines Objektes in SMIv2

Die einzelnen Parameter des Makros OBJEKT-TYPE bedeuten dabei das folgende:

- **SYNTAX**
  Dieser Parameter definiert den Datentyp des Objektes. Datentypen und Textkonventionen die hier aufgeführt werden können sind zum Beispiel in den Tabellen 4.5 und 4.6 aufgelistet.

- **MAX-ACCESS**
  Enthält Informationen über die Zugriffsrechte auf das Objekt, ob es zum Beispiel nur lesbar oder auch beschreibbar ist oder bei Tabellen neue Zeilen in die Tabelle eingefügt werden dürfen. Gültige Zugriffsrechte in SMIv2 sind:
  - read-only, read-write, read-create, accessible-for-notify, not-accessible.
  SMIv1 kennt stattdessen den Parameter ACCESS, der sich bei SMIv2 in MIN-ACCESS und MAX-ACCESS aufteilt, wobei MIN-ACCESS eine optionale Zusatzangabe über die minimalen Zugriffsrechte des Objektes ist, die bei fehlender Angabe mit den MAX-ACCESS Werten gleichgesetzt werden.

- **STATUS**
  Ein Agent sollte immer mindestens all die Objekte eines MIB-Modules unterstützen, die den Status CURRENT innehaben.
SMIV1 kennt die Optionen MANDATORY, OPTIONAL und OBSOLETE. Wobei ein Agent zwingend alle mit MANDATORY gekennzeichneten Objekte unterstützen sollte.

• DESCRIPTION

Dieser letzte zwingende Parameter enthält eine Beschreibung darüber, was das Objekt, bzw. dessen Managementinformation macht. Dieser Parameter dient ausschließlich dem Menschen, da es ihm die Funktion des Objektes erläutert.

Über diese vier zwingende Parameter hinaus gibt es noch weitere, optionale Parameter, mit denen eine recht detaillierte Definition von Managementinformationen über Objekte möglich ist.

Im folgenden ist ein Auszug der „SNMPv2-MIB“-Datei beispielhaft für die SMI-Objekt-Definition aufgeführt.

```
SNMPv2-MIB DEFINITIONS ::= BEGIN

IMPORTS
MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
TimeTicks, Counter32, snmpModules, mib-2
FROM SNMPv2-SMI
DisplayString, TestAndIncr, TimeStamp
FROM SNMPv2-TC
MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
FROM SNMPv2-CONF;

snmpMIB MODULE-IDENTITY
LAST-UPDATED "200210160000Z"
ORGANIZATION "IETF SNMPv3 Working Group"
CONTACT-INFO "..."
[...]
 ::= { snmpModules 1 }

snmpMIBObjects OBJECT IDENTIFIER ::= { snmpMIB 1 }
[ . . . ]
-- the System group
--
-- a collection of objects common to all managed systems.

system OBJECT IDENTIFIER ::= { mib-2 1 }

sysDescr OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..255))
MAX-ACCESS read-only
STATUS   current
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."
 ::= { system 1 }
[...]
```

Listing 4.7: Auszug aus der SNMPv2-MIB (Entnommen RFC 3418)

In der Datei SNMPv2-MIB sind Objekte zu finden, die Managementinformationen über die Systemumgebung liefern, in der die SNMP-Agentenanwendung läuft. Diese Objekte sind in der Gruppe „system“ zusammengefasst. Unter der OID „iso.org.dod.internet.mgmt.mib-2.system.sysDescr“ bzw. „1.3.6.1.2.1.1.1“ ist das Objekt „sysDescr“ zu finden, welches eine textuelle Beschreibung der Agenten-Systemumgebung definiert. Um diese Beschreibung selbst abzurufen muss an das Ende der OID des Objektes noch ein Instanzwert angehängt werden, so dass die eigentliche Managementinformationsansteuerung die Form OID.y hat. Bei einfach strukturierten Objekten ist die angehängte Instanz für gewöhnlich die Zahl 0, so dass hier in diesem Fall die Beschreibung des Systems unter der OID und der Instanz „1.3.6.1.2.1.1.1.0“ zu finden ist. Weitere Instanzen besitzt ein solches Objekt nicht. Bei Objekten die zu Tabellen strukturiert sind, hängt die Instanz von der Stelle der Managementinformation in der Struktur ab. Hier muss zwischen den verschiedenen Instanzen unterschieden werden, was beim Betrachten der Definition von Tabellen deutlich wird.
Listing 4.8: Auszug der Definition der Tabelle „ifTable“ (Entnommen RFC 2863 - The Interfaces Group)
Das Beispiel in Listing 4.8 zeigt einen Auszug des Objektes „ifTable“, das in der Gruppe „interfaces“ innerhalb der IF-MIB definiert ist. Das Objekt „ifTable“ ist als Tabelle strukturiert und enthält viele weitere Objekte die Eigenschaften und Statusinformationen von Netzwerkschnittstellen definieren. In der praktischen Anwendung ist dann für jede vorhandene Netzwerkschnittstelle des Gerätes eine Zeile in der „ifTable“ belegt, deren Inhalt über die Instanz angesteuert werden kann. Das Objekt „ifTable“ ist definiert als eine Liste über „ifEntry“-Objekte. Das Objekt „ifEntry“ wiederum ist definiert als eine Liste über verschiedene Objekte, die die eigentlichen Managementinformationen einer Netzwerkschnittstelle definieren. Jeder Zelle der Tabelle ist somit eine OID zugeordnet, über die die enthaltene Managementinformation angesteuert werden kann.

SMIv2 erlaubt auch die Definition von Ereignismeldungen. Diese sind dann ebenfalls in der „Management Information Base“ (MIB) abgelegt und mit einer OID versehen, die sie eindeutig Kennzeichnen. In dem folgenden Beispiel wird eine Meldung definiert, die den Manager darüber aufklärt, dass eine Netzwerkschnittstelle in der Systemumgebung des Agenten aktiviert wurde.

Wird die Netzwerkschnittstelle durch den Agenten überwacht, so sendet er dem Manager, nach Aktivierung der Schnittstelle, die OID des Notification-Objektes und eventuell auch weitere beliebige Managementinformationen über eine Ereignismeldung zu. Der Manager kann dann über die empfangene OID die zugehörige Notification-Objekt-Definition aus seiner eigenen MIB für weitere Erläuterungen herausfinden.

Im Kapitel 6 finden sich einige praktische Beispiele im Bezug zur Software Net-SNMP, die die praktische Anwendung von SMIv2 und die Funktion der Objekte noch etwas verdeutlichen.
4.3.3.3 Die Management Information Base


Einige Gruppen der MIB-II sind hier nun aufgezählt.

<table>
<thead>
<tr>
<th>Gruppe (OID)</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>system (1)</td>
<td>Diese Gruppe muss auf jedem SNMP Agenten vorhanden sein. Sie beinhaltet Informationen über den Agenten selbst wie zum Beispiel Betriebszeit, Systemname und -beschreibung, usw.</td>
</tr>
<tr>
<td>interfaces (2)</td>
<td>Beinhaltet die Anzahl und Beschreibung der Netzwerkschnittstellen des Agenten, sowie Status- und Verkehrslowsinformationen</td>
</tr>
<tr>
<td>at (3)</td>
<td>Zeigt die Adressübersetzung an (zum Beispiel von Ethernet- in IP-Adressen)</td>
</tr>
<tr>
<td>ip (4)</td>
<td>Beinhaltet Tabellen, die die IP-Adressen sowie IP-Routing Informationen des Agenten anzeigen, desweiteren Informationen über die IP-Paketstatistik, zum Beispiel die Anzahl der verworfenen IP-Pakete.</td>
</tr>
<tr>
<td>icmp (5)</td>
<td>Enthält eine Statistik über empfangene und gesendete ICMP Nachrichten-Pakete</td>
</tr>
<tr>
<td>tcp (6)</td>
<td>Die TCP Gruppe enthält Managementinformationen, die den TCP-Algorithmus und seine Parameter betreffen. Hier wird die aktuelle und kumulierte Anzahl von offenen Verbindungen überwacht</td>
</tr>
<tr>
<td>egp (8)</td>
<td>Die Verkehrssstatistik über das externe Gateway-Protokoll. Dient Routern die EGP unterstützen.</td>
</tr>
<tr>
<td>cmot (9)</td>
<td>Wird nicht mehr verwendet und ist hier nur der Vollständigkeit halber aufgeführt. CMOT ist der Managementstandard CMIP des OSI-Management-Modells für TCP/IP</td>
</tr>
<tr>
<td>transmission (10)</td>
<td>Diese Gruppe ist ein Platzhalter für medienspezifische Objekte. Hier können zum Beispiel ethernetspezifische Statistiken gefunden werden</td>
</tr>
</tbody>
</table>

Tabelle 4.8: Einige Gruppen der MIB-II

4.3.4 Das Netzwerkmanagement-Protokoll SNMP

Das Netzwerkmanagement-Protokoll SNMP dient der Kommunikation zwischen dem Manager und seinen Agenten. Üblicherweise sendet der Manager dem Agenten eine Anfrage und bittet diesen um Managementinformationen. Der Manager kann dem Agenten auch neue Informationen zusenden, die dieser in seinem System übernehmen soll. Der Agent antwortet mit der angeforderten Managementinformation oder bestätigt, dass er seinen Zustand wie angefordert aktualisiert hat. Der Agent kann auch bei Eintritt eines bestimmten Ereignisses unaufgefordert dem Manager Informationen zusenden. Daneben können auch verschiedene Fehler gemeldet werden, zum Beispiel „notWritable“ um anzuzeigen, dass eine Managementinformation nur gelesen werden kann und nicht wie versucht, neu geschrieben werden darf. Für die Übertragung werden die Managementinformationen mit den ASN.1 Kodierungsregeln BER kodiert.

Bezüglich der Verwendung von SNMP können die Nachrichten des Netzwerkmanagement-Protokolls in drei Gruppen unterteilt werden.

- **Befehl (Request)**
  - Nachricht vom Manager an den Agenten
- **Antwort (Response)**
  - Antwort des Agenten auf einen Befehl des Managers
- **Meldung (Trap/Inform)**
  - unaufgeforderte Nachricht des Agenten an den Manager

Der Aufbau des Netzwerkmanagement-Protokolls SNMP ist recht einfach gehalten und kann in drei Teile untergliedert werden.

1) verwendete SNMP Version
2) Informationen bzgl. des verwendeten Sicherheitsmodells
3) Protocol Data Unit (SNMP-PDU)


Im folgenden werden nun die verschiedenen SNMP-Nachrichtentypen vorgestellt, die für die Kommunikation zwischen SNMP-Manager und SNMP-Agent verwendet werden können.

- **Response**
  - Die Response Nachricht ist die Antwort, die meist vom Agenten an den Manager nach Erhalt eines Befehls gesendet wird. Sie beinhaltet die OIDs und die Managementinformationen der vom Manager zuvor angesprochenen Objekte. Ist eine Managementinformation nicht verfügbar, so beinhaltet die Response Nachricht eine der in SNMP standardisierten Fehlerrmeldungen.
  - Sendet der Agent dem Manager eine InformRequest Nachricht, so bestätigt der Manager diese dem Agenten ebenfalls mit einer Response Nachricht.

- **GET**
  - Mit der GetRequest Nachricht fordert der Manager den Agenten auf, Managementinformationen auszulesen und zu übertragen. Neben der OID des zur Managementinformation gehörigen Objektes muss der Manager auch die Instanznummer der Managementinformation kennen. Der Manager kann über eine GetRequest Nachricht auch mehrere Managementinformationen auf einmal abfragen.
• **GETNEXT**

• **GETBULK**
  Die GetBulkRequest Nachricht wurde erst mit SNMPv2 eingeführt um die Effizienz von SNMP, vor allem im Umgang mit Tabellen, zu steigern. GetBulkRequest baut auf GetNextRequest (-PDUs) Nachrichten auf. Für die Abwärtskompatibilität zu SNMPv1 wird allerdings ein Protokollumwandler benötigt.
  Bei der GetBulkRequest Nachricht muss der Manager zusätzlich, zu den OIDs der Objekte deren Managementinformation er abfragen möchte, noch zwei Integervariablen mit übersenden. Die erste Integervariable „non-repeaters” gibt die Anzahl der angefragten OIDs auf der übersandten Abfrage-OID-Liste an, die mit einer gewöhnlichen GetNextRequest-PDU angesprochen werden. Die zweite Integervariable „max-repetitions” gibt die Anzahl der GetNextRequest-PDUs an, die dann auf alle verbleibenden OIDs auf der übersandten Abfrage-OID-Liste angewandt werden. Die GetBulkRequest Nachricht ist Dank der beiden Integervariablen gut dafür geeignet Managementinformationen aus Tabellen auszulesen. Die Objekte der Tabelle, die in einer Sequenz strukturiert sind und die Spalten der Tabelle darstellen, und die verschiedenen Instanzen der Objekte, die den Zeilen der Tabelle entsprechen, können so zusammenhängend angesprochen werden.

• **SET**
  Die SetRequest Nachricht ist die einzige Schreiboperation die SNMP bietet. Sie benötigt die OID des Objektes und die Instanz sowie natürlich die neu zu setzende Managementinformation. Außerdem muss auch der Datentyp angegeben werden. Der Agent bestätigt die Übernahme der Managementinformation und die Aktualisierung seines Zustandes mit einer Response Nachricht, die die aktuelle Managementinformation des Objektes enthält. Treten Fehler während der Aktualisierung auf, antwortet der Agent mit einer der in SNMP standardisierten Fehlermeldungen.

• **SNMPv1-Trap und SNMPv2-Trap**

- **Inform**


- **Report**


Dies sind alle Nachrichtentypen, die die Standard-Spezifikation von SNMP definiert.

Wie bereits erwähnt, enthält der als SNMP-PDU bezeichnete Teil des Protokolls die verschiedenen Nachrichtentypen, die das Netzwerkmanagement-Protokoll SNMP bietet.
Die SNMP-PDU ist dabei für die einzelnen SNMP-Nachrichtentypen wie folgt aufgebaut:

<table>
<thead>
<tr>
<th>SNMPv1-Trap</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PDU-Type</strong></td>
</tr>
<tr>
<td><strong>variable-bindings</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNMPv2-Trap</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PDU-Type</strong></td>
</tr>
<tr>
<td><strong>variable-bindings</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InformRequest</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PDU-Type</strong></td>
</tr>
<tr>
<td><strong>variable-bindings</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Die PDU-Typen sind folgendermaßen aufgebaut:

- **GetRequest, GetNextRequest, SetRequest, SNMPv2-Trap, InformRequest:**
  - **PDU-Type** | **request-id** | **0** | **0** | **variable-bindings**
  - **Response:**
  - **PDU-Type** | **request-id** | **error-status** | **error-index** | **variable-bindings**
  - **GetBulkRequest**
  - **PDU-Type** | **request-id** | **non-repeaters** | **max-repetitions** | **variable-bindings**
  - **SNMPv1-Trap**
  - **PDU-Type** | **enterprise** | **address** | **specific** | **timestamp** | **variable-bindings**
  - **variable-bindings**
  
<table>
<thead>
<tr>
<th>Object ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Abb. 4.19: Aufbau der SNMP-PDU

Im folgenden werden die vier Felder einer (Response) SNMP-PDU erklärt:

- **request-id**
  Der Wert dieses Feldes einer Response Nachricht muss dem Wert des Feldes der dazugehörigen Request Nachricht entsprechen. Der Manager kann so die Antworten eines Agenten den einzelnen, zuvor von ihm gesendeten Befehlen, zuordnen.

- **error-status**

<table>
<thead>
<tr>
<th>Fehlermeldung (Code)</th>
<th>GetRequest</th>
<th>GetNextRequest</th>
<th>SetRequest</th>
<th>InformRequest</th>
</tr>
</thead>
<tbody>
<tr>
<td>noError (0) (auch SNMPv1)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>tooBig (1) (auch SNMPv1)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>noSuchName (2) (nur SNMPv1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>badValue (3) (nur SNMPv1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readOnly (4) (nur SNMPv1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>genError (5) (auch SNMPv1)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>noAccess (6)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wrongType (7)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wrongLength (8)</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wrongEncoding (9)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>wrongValue (10)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>noCreation (11)</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inconsistentValue (12)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>resourceUnavailable (13)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>commitFailed (14)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>undoFailed (15)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>authorizationError (16)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>notWritable (17)</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inconsistentName (18)</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 4.10: „error-status“ Codes der Response Nachricht
• **error-index**
  Wenn das „error-status“ Feld nicht leer ist, identifiziert der „error-index“ das Objekt in der Objekt-Varialben-Liste, das den Fehler verursacht. Verursacht das erste Objekt in der Liste den Fehler, so ist der Index 1, das zweite Objekt hat den Index 2, usw. .

• **variable-bindings**
  Die Objekt-Varialben-Liste ist eine Liste aus Paar-Elementen. Das erste Element ist die OID des referenzierten Objektes. Das zweite Element entspricht einem der folgenden Werte:
  - **value**: Datentyp und zugehörige Managementinformation des referenzierten Objektes
  - **unSpecified**: Null-Wert, wird in den Lese-Anfragen des Managers verwendet.
  - **noSuchObject**: Zeigt an, dass das Objekt im Agenten nicht implementiert ist
  - **noSuchInstance**: Zeigt an, dass die Instanz des Objektes für die Operation nicht gültig ist
  - **endOfMibView**: Zeigt an, dass das Objekt außerhalb der (zugriffsberechtigten) MIB liegt

Der Aufbau der kompletten SNMP-Nachricht unterscheidet sich zwischen den Versionen. Die SNMP Versionen SNMPv1 und -v2c besitzen neben der SNMP-PDU lediglich noch die beiden Felder für die Angabe der Version und dem Community-Namen.

Der Aufbau der kompletten Nachricht dieser Versionen sieht wie folgt aus.

<table>
<thead>
<tr>
<th>msgVersion</th>
<th>community</th>
<th>SNMP-PDU</th>
</tr>
</thead>
</table>

Abb. 4.20: Aufbau der SNMPv1 und -v2c Nachrichten

SNMPv3 Nachrichten zeichnen sich durch einen komplexeren Aufbau aus, der dem besseren Sicherheitsmodell und dem modularen Aufbau der SNMPv3 Kommunikationsteilnehmer geschuldet ist.

<table>
<thead>
<tr>
<th>msgVersion</th>
<th>msgID</th>
<th>msgMaxSize</th>
<th>msgSecurityModel</th>
<th>$\ldots$</th>
<th>contextEngineID</th>
<th>contextName</th>
<th>SNMP-PDU</th>
</tr>
</thead>
</table>

Abb. 4.21: Aufbau der SNMPv3 Nachrichten


Die SNMP-Kommunikation mit Hilfe von Net-SNMP ist im folgenden dargestellt:

Um1 # snmpget -v1 -c public 10.0.0.2.1.3.6.1.2.1.2.2.1.2.1.1.3.6.1.2.1.2.2.1.10.1
IF-MIB::ifDescr.1 = STRING: "eth1"
IF-MIB::ifInOctets.1 = Counter32: 3319

Befehl 4.1: SNMPv1 Anfrage, ausgeführt mit Net-SNMP

Hier ist nun die entschlüsselte Antwort (Response) des Agenten auf die Anfrage des Managers.

Abb. 4.22: Entschlüsselung einer SNMPv1 Response Nachricht
4.4 Sicherheit in SNMP


4.4.1 SNMPv1 und SNMPv2c

SNMPv1 und SNMPv2c nutzen ein Community-basierendes Administration Framework für die Sicherheit der Kommunikation zwischen den verschiedenen Komponenten des SNMP Netzwerkes. Eine Community besteht dabei aus mindestens einem Agenten und einem Manager. Der Community-Name ist eine ASCII-Zeichenkette und Bestandteil jeder SNMPv1 und SNMPv2c Nachricht. Er dient der Authentifizierung. Der Manager muss in jeder Anfrage an den Agenten dessen Community-Namen angeben. Besitzt der Agent den Community-Namen nicht, so antwortet er nicht auf die Anfrage. Es können mehrere Communities in einem SNMP-Agenten eingerichtet werden. Unterschiedlichen Community-Namen können zudem unterschiedliche Zugriffsrechte zugeordnet werden. Dies ermöglicht die Vergabe individueller Zugangsberechtigungen für bestimmte Teilbereiche des verwalteten Netzwerks. Die Community besitzt somit die Funktion eines Zugangskontos, wobei der Community-Name auch gleichzeitig das Passwort des Kontos darstellt. In der einfachsten und weit verbreitetsten Form gibt es lediglich zwei Community-Zugriffsberechtigungen:

- **read-only** - erlaubt nur lesenden Zugriff
- **read-write** - erlaubt auch das konfigurieren von Parametern


4.4.2 SNMPv3

Mit der Einführung von SNMPv3 sollte die unzureichende Sicherheitsfunktionalität in SNMP beseitigt werden. Vor allem die folgenden Aspekte waren bezüglich des Sicherheitskonzeptes zu erfüllen:

- Verhinderung der Veränderung von Managementinformationen durch Unbeteiligte
- Verhinderung der Einspeisung von falschen oder veränderten Nachrichten durch Unbeteiligte
- Verhinderung von falschen Manager-Identitäten
- Optionale Verschlüsselung der Kommunikation

SNMPv3 bietet die Möglichkeit Authentifizierungs- und Verschlüsselungstechniken einzubeziehen. Vor allem aufgrund der modularen Architektur von SNMPv3 können dabei unterschiedlich realisierte Sicherheitstechniken verwendet werden. Im Rahmen der SNMPv3 Entwicklung wurden zwei Sicherheitsmodelle spezifiziert, das „View-Based Access Control Model“ und das „User-Based Security Model“, die im folgenden näher erläutert werden.

4.4.2.1 Das User-Based Security Model


4.4.2.2 Das View-Based Access Control Model

Das „View-Based Access Control Model“ beschreibt Elemente der Zugriffskontrolle auf die Managementinformationen eines Agenten über die OID der Objekte. Mit Hilfe des Models kann bestimmt werden, wer in welcher Form auf welche Managementinformationen Zugriff hat. Das gegenwärtige Model wird im RFC 3415 (STD 62) beschrieben. In diesem RFC wird auch ein standardisiertes MIB-Modul, das SNMP-VIEW-BASED-ACM-MIB Modul, spezifiziert, dessen Objekte die Verwaltung der Parameter via SNMP ermöglichen. VACM besitzt die folgenden Konfigurationsparameter, die in einem SNMPv3 Agenten konfiguriert werden können:

• **Groups**
  Unter diesem Parameter können Zugriffsrechte unter einem Gruppennamen zusammengefasst werden. Dabei bezieht sich eine Gruppe auf eine Menge von Tupeln der Form <securityModel, securityName>. Umgekehrt kann eine Tupelkombination höchstens einer Gruppe zugeordnet werden. Der Begriff securityModel steht dabei für das verwendete Sicherheitsmodell, zum Beispiel SNMPv1, SNMPv2c oder USM. Der Begriff securityName steht für eine eindeutige Kennung.

• **securityLevel**
  Das securityLevel definiert die Sicherheitsstufe, die beim Zugriff auf ein Objekt aktiviert ist. So kann zum Beispiel ein Schreibzugriff auf ein Objekt eine Authentifizierung erfordern, während der Lesezugriff ohne erlaubt ist. Für gewöhnlich können drei verschiedene Sicherheitsstufen definiert werden, die sich auf die Authentifizierung und die Verschlüsselung beziehen.
  - noAuthNoPriv - keine Sicherheitsfunktionen sind aktiviert
  - authNoPriv - Authentifizierung über ein Passwort wird verlangt
  - authPriv - sowohl die Authentifizierung als auch die Verschlüsselung wird verlangt
  Verschlüsselung ohne Authentifizierung wurde für nicht sinnvoll befunden, daher gibt es den securityLevel „noAuthPriv“ nicht.

• **Contexts**

• **MIB Views**
  MIB Views sind Sichten auf die MIB. Dabei wird ein bestimmter Ausschnitt auf die MIB definiert, der dann einer Gruppe zugeordnet werden kann. So kann recht einfach der Zugriff einer Gruppe auf die Objekte eines bestimmten Ausschnitts der MIB geregelt werden.

• **Access Policy**
  Über die Access Policy wird schließlich die eigentliche Zugriffskontrolle definiert. Es fasst die obigen Parameter zusammen und definiert so den Zugriff auf die Managementinformationen des Agenten. Somit muss zum Beispiel eine MIB View nur einmal definiert werden und kann dann verschiedenen Gruppen mit unterschiedlichen Zugriffsrechten zugeteilt werden.

Die SNMP-Software Net-SNMP unterstützt VACM. Ein praktisches Beispiel für die Konfiguration von VACM in einem Net-SNMP Agenten ist im Kapitel 6.5.4.1 zu finden.
4.5 Erweiterungsmodelle für den SNMP-Agenten

Um auf Managementinformationen zugreifen zu können benötigt ein SNMP Agent entsprechende Schnittstellen zu den Ressourcen, die diese Managementinformationen stellen. Diese Schnittstellen können und müssen nicht immer Teil der Implementierung des Agenten selbst sein.

Ein Problem ergibt sich vor allem, wenn mehrere voneinander unabhängige Anwendungsprogramme auf einem netzwerkfähigen Gerät über eine Netzwerkschnittstelle Managementinformationen via SNMP bereitstellen wollen. Diese können nicht alle parallel einen eigenen SNMP-Agenten betreiben, da auf einem netzwerkfähigen Gerät für gewöhnlich nur ein einziger SNMP-Agent bereitstehen sollte, mit dem ein Manager interagieren kann. Um die Managementinformationen der Anwendungsprogramme trotzdem via SNMP einem Manager verfügbar zu machen, benötigt es in diesem Fall einen SNMP Master-Agenten, über den der Master-Agent die vom Manager kommenden SNMP-Nachrichten ausleitet, entsprechend den angesteuerten Managementinformationen, an den jeweiligen Sub-Agenten in einem speziellen Protokollformat weiter.


4.5.1 SMUX


4.5.2 AgentX


4.6 Zusammenfassung

In diesem Kapitel wurde der Aufbau und die Funktionsweise von SNMP ausführlich vorgestellt. Die theoretischen Grundlagen von SNMP, die in diesem Kapitel beschrieben sind, erleichtern das Verständnis für die Funktion der SNMP-Implementierung Net-SNMP, die im weiteren Verlauf der Arbeit noch vorgestellt wird, und zeigen auch die grundsätzlichen Möglichkeiten des Einsatzes von SNMP im Netzwerkmanagement auf.
5 Netzwerk-Simulationen mit Virtual Network User Mode Linux


5.1 Einleitung


Das VNUMUL-Projekt ist im Internet unter der Adresse „http://jungla.dit.upm.es/~vnumul“ zu finden.


5.2 VNUMUL-Grundlagen

Im folgenden werden nun die Komponenten von VNUMUL und einige Begrifflichkeiten erläutert. Die Begriffe werden von den VNUMUL-Autoren selbst und auch im weiteren Verlauf dieser Arbeit verwendet.

- **Host (Host-System)**

- **UML-Rechner (virtuelle Maschine)**
Virtuelles Netzwerk


Szenario


Simulation


Management Netzwerk


Virtuelle Netzwerkschnittstelle


Im Bezug auf die Erstellung eines Szenarios und die Durchführung der Simulation heben die Autoren von VNUML zwei Komponenten hervor.

• **VNUML-Sprache**

• **VNUML-Parser**

Um eine Simulation über einem Szenario erfolgreich zu starten, muss die VNUML-Sprache, in der das Szenario beschrieben ist, die gleiche Version besitzen, wie der VNUML-Parser, mit dem die Simulation durchgeführt wird.

VNUML verwendet UML-Rechner zum Aufbau der Netzwerk-Simulationen. UML-Rechner funktionieren auch unabhängig von VNUML und können zu virtuellen Netzwerken zusammengeschlossen werden. VNUML versteckt jedoch die komplexen Arbeitsschritte, die dafür notwendig sind, hinter seiner relativ einfachen Beschreibungssprache und macht so den Einsatz von UML-Rechnern für immer wieder rekonstruierbare Netzwerk-Simulationen schnell, dynamisch, überschaubar und komfortabel.

UML-Rechner bestehen aus zwei Komponenten die auch VNUML für seine Funktion benötigt und die in jedem Netzwerk-Szenario explizit angegeben werden müssen.

• **User Mode Linux Kernel (UML-Kernel)**

• **User Mode Linux Root-Dateisystem (UML-Root-Dateisystem)**
Verschiedene UML-Root-Dateisysteme für die unterschiedlichen VNULM-Versionen zum durchspielen der Beispiel-Szenarios stehen auf der VNULM Internetseite zum Download bereit. Diese basieren auf der Linux Distribution „Debian“ und können weitestgehend beliebig modifiziert und erweitert werden.


Allgemeine Hinweise zur Installation von Anwendungsprogrammen in die UML-Rechner werden im Kapitel 5.6, zur Installation der Routing-Software Quagga im Kapitel 5.7 und zur Installation der SNMP-Software NetSNMP in Kapitel 6.3, behandelt. Eine Anleitung zum Erstellen eines UML-Rechners auf Basis der Linux-Distribution Gentoo ist im Anhang dieser Ausarbeitung, im Kapitel 7, zu finden.

5.3 Durchführung einer VNULM Simulation

Im folgenden wird nun der Aufbau eines Szenarios und die Durchführung einer Simulation an einem einfachen Beispiel beschrieben. Die VNULM Autoren empfehlen die Durchführung einer Netzwerk-Simulation in drei Phasen zu unterteilen.

- **Entwurfs Phase**

- **Implementierungs Phase**

- **Ausführungs Phase**


3. Stoppen der Simulation: Über die Befehlsoption `-d` stoppt der VNUML-Parser die Simulation wieder. Die einzelnen UML-Rechner werden heruntergefahren und die virtuellen Netzwerkschnittstellen werden entfernt. Dabei wird eine Simulation so gestoppt, das die gegenwärtige Konfiguration der einzelnen UML-Rechner für den nächsten Start erhalten bleibt. Über die Befehlsoption `-P` wird die Simulation abgebrochen und all ihre Backup-Dateien werden gelöscht, so dass sie sich beim nächsten Start wieder im ursprünglichen Ausgangszustand befindet.


5.3.1 Entwurfs Phase

Die Topologie des Netzes sieht somit wie folgt aus:

![Diagramm der Netzwerktopologie](image)

\[Abb. 5.2: Topologie des Netzwerk-Szenarios „SimpleNet“\]


In dieser Abbildung nicht zu sehen ist das Management-Netzwerk, das der VNUML-Parser zwischen dem Host und den einzelnen UML-Rechnern automatisch aufbaut um diese per SSH während einer Simulation zu verwalten.

### 5.3.2 Implementierungs Phase


Die Syntax der VNUML-Sprache, die Tags, werden von den VNUML-Autoren in drei Typen unterteilt.

- **Strukturelle Tags**

- **Topologie Tags**
  Diese Tags beschreiben die Topologie des Szenarios. Der VNUML-Parser verwendet sie während des Startens und Stoppens der Netzwerk-Simulation und ignoriert sie während der Ausführungs Phase. Zu den Topologie Tags gehören <kernel>, <filesystem>, <conf0>, <xterm>, <ssh_key>, <automac>, <ip>, <ipv4>, <forwarding>, <route>, <host_mapping> und noch einige mehr.

- **Simulations Tags**

Zusätzlich gibt es noch Tags wie zum Beispiel <simulation_name> und <version>, die zu den Simulations sowie auch zu den Topologie Tags gehören.

Die Beschreibung eines Szenarios in der XML-Datei beginnt xml-typisch mit der Angabe der verwendeten XML-Version und der Kodierung sowie der Angabe in welcher Datei die Definition der VNUML-XML-Tags zu finden ist. Die Struktur der XML-Tags der VNUML-Sprache ist in Document Type Definition (DTD) in der

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.5</version>
    <simulation_name>SimpleNet</simulation_name>
    <ssh_key>/root/.ssh/identity.pub</ssh_key>
    <automac/>
    <host_mapping/>
    <default_filesystem type="cow">'/DirPath/UML-Root-Filesystem'</default_filesystem>
    <default_kernel>'/DirPath/UML-Kernel'</default_kernel>
  </global>
```

Listing 5.1.1: Inhalt der Datei SimpleNet.xml, Teil 1


```
<net name="Net0" mode="virtual_bridge" />
<net name="Net1" mode="virtual_bridge" />
```

Listing 5.1.2: Inhalt der Datei SimpleNet.xml, Teil 2

Netzwerk-Simulationen mit Virtual Network User Mode Linux


Listing 5.1.3: Inhalt der Datei SimpleNet.xml, Teil 3

```xml
<vm name="uml1">
  <boot><con0>xterm</con0></boot>
  <if id="1" net="Net0">
    <ipv4 mask="255.255.255.252" >10.0.0.1</ipv4>
  </if>
  <route type="inet" gw="10.0.0.2">default</route>
  <filetree root="/etc/snmp" when="start">/'UML1ConfigfilesDir'/snmp</filetree>
  <exec seq="startSNMP" type="verbatim">snmpd -c /etc/snmp/snmpd.conf</exec>
  <exec seq="stopSNMP" type="verbatim">killall snmpx</exec>
  <exec seq="readLogs" type="verbatim">cat /var/log/snmpd.log</exec>
  <exec seq="mgmt" type="verbatim">snmpwalk -v1 -cpublic 10.0.0.2 .1.3.6.1.2</exec>
</vm>

<vm name="uml2">
  <boot><con0>xterm</con0></boot>
  <if id="1" net="Net0">
    <ipv4 mask="255.255.255.252" >10.0.0.2</ipv4>
  </if>
  <if id="2" net="Net1">
    <ipv4 mask="255.255.255.252" >164.1.2.1</ipv4>
  </if>
  <forwarding type="ip"/>
  <filetree root="/etc/snmp" when="start">/'UML2ConfigfilesDir'/snmp</filetree>
  <filetree root="/etc/quagga" when="start">/'UML2ConfigfilesDir'/quagga</filetree>
  <exec seq="startSNMP" type="verbatim">snmpd -c /etc/snmp/snmpd.conf</exec>
  <exec seq="stopSNMP" type="verbatim">killall snmpd</exec>
  <exec seq="startRouter" type="verbatim">zebra -f /etc/quagga/zebra.conf -d</exec>
  <exec seq="stopRouter" type="verbatim">killall zebra</exec>
  <exec seq="readLogs" type="verbatim">cat /var/log/zebra.log</exec>
  <exec seq="readLogs" type="verbatim">cat /var/log/zebra.log</exec>
</vm>

<vm name="uml3">
  <boot><con0>xterm</con0></boot>
  <if id="1" net="Net1">
    <ipv4 mask="255.255.255.252" >164.1.2.2</ipv4>
  </if>
  <route type="inet" gw="164.1.2.1">default</route>
</vm>
```

Damit ist die Implementierung des Szenarios „SimpleNet“ abgeschlossen und die Ausführungs Phase mit der Durchführung der Netzwerk-Simulation kann anfangen werden.

- 49 -
5.3.3 Ausführungs Phase


```
host # vnumlparser.pl -t SimpleNet.xml -vB
```


Das komplette Netzwerk, inklusive Management-Netzwerk, das der VNUML-Parser für das Beispiel „SimpleNet“ aufbaut, sieht wie folgt aus.

```
192.168.0.2
192.168.0.0/30
eth1

192.168.0.1
192.168.0.0/30
eth0

192.168.0.9
192.168.0.2
192.168.0.0/30
192.168.0.4/30
eth1
eth0

192.168.0.5
192.168.0.10
164.1.2.0/30
eth1
eth0

164.1.2.1
164.1.2.2
164.1.2.0/30
eth1
eth0

UML 1
UML 2
UML 3

eth0
eth0
eth0
```

```
Abb. 5.3: Komplette virtuelle Netzwerk-Topologie der VNUML Simulation des „SimpleNet“-Szenarios
```


Das Simulation-Kommando „start“ des Beispiels kopiert die Quagga und SNMP Konfigurationsdateien auf die jeweiligen UML-Rechner.

```
host # vnumlparser.pl -x start@SimpleNet.xml -v
```

```
host # vnumlparser.pl -x startSNMP@SimpleNet.xml
host # vnumlparser.pl -x startRouter@SimpleNet.xml
```

```
Befehl 5.4: Ausschnitt der Ausgabe des Simulations-Kommandos „mgmt“
```

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux [VNUML Name] 2.6.12.6-1m #5 Mon Jul 3 23:32:43 UTC 2006 i686
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::odu.0.0.0.0.0.0.0
SNMPv2-MIB::sysUpTime.0 = Timeticks: (24358) 0:04:03.58
[..]
```

```
Befehl 5.4: Ausschnitt der Ausgabe des Simulations-Kommandos „mgmt“
```

5 Netzwerk-Simulationen mit Virtual Network User Mode Linux
Wurde ein Szenario mit seinen Simulations-Kommandos fertig durchgespielt und soll die Netzwerk-Simulation beendet werden, kann sie über die folgende Befehlsoption des VNUML-Parsers gestoppt werden.

```
host # vnumlparser.pl -d SimpleNet.xml -VB
```

Befehl 5.5: Beenden einer Netzwerk-Simulation

Nach dem Stoppen der Simulation behält VNUML allerdings noch einige Backup-Dateien, die es zum Beispiel erlauben die beteiligten UML-Rechner und somit den letzten Zustand der Simulation zu speichern. Treten Fehler auf oder soll eine Simulation von neuem beginnen und in den Ausgangszustand versetzt werden, so kennt VNUML die Befehlsoption `-P`, die alle Backup-Dateien löscht und auch eine laufende Netzwerk-Simulation zum Abbruch zwingt.

```
host # vnumlparser.pl -P SimpleNet.xml -v
```

Befehl 5.6: Abbruch einer Netzwerk-Simulation und Löschen aller Backup-Dateien

Die einfache Eingabe des Befehls „vnumlparser.pl“ listet alle möglichen Befehlsarten mit einer kurzen Erläuterung auf.

### 5.4 Das VNUML Management-Netzwerk


Der folgende Befehl führt vom Host aus eine Verfolgung der Route von „uml1“ zu „uml3“ durch.

```
host # ssh root@192.168.0.2 'tracepath 164.1.2.2'
```

```
root@192.168.0.2's password: xxxx
1: uml1 (10.0.0.1) 2.050ms pmtu 1500
1: 10.0.0.2 (10.0.0.2) 0.977ms
2: 164.1.2.2 (164.1.2.2) 1.049ms reached
    Resume: pmtu 1500 hops 2 back 2
```

Befehl 5.7: Routenverfolgung von „uml1“ zu „uml3“, vom Host via „ssh“ aus initiiert

Der Befehl „tracepath 164.1.2.2“ wird via SSH an den UML-Rechner „uml1“ gesendet, dort ausgeführt und das Ergebnis wieder via SSH an den Host gesendet und dort auf der Konsole ausgegeben. Mit SSH allein, ohne VNUML ist so auch eine Konfiguration der UML-Rechner vom Host aus möglich, ohne die Szenario Beschreibung ändern zu müssen.


```
host # scp UML2Configs/quagga/ripd.conf root@192.168.0.6:/etc/quagga/
```

```
root@192.168.0.6's password: xxxx
100% 293 0.3KB/s 00:00
```

Befehl 5.8: Kopieren der Datei „ripd.conf“ auf den UML-Rechner „uml2“ mittels „scp“


Wird SNMP nicht nur in den UML-Rechnern, sondern auch auf dem Host verfügbar gemacht, so kann es als Erweiterung des VNUML-Managements dienen, das vor allem die Bereiche des Leistungs- und des Fehlermanagements sehr viel besser abdeckt. Zumal es einige auf SNMP Software wie Net-SNMP aufbauende Management-Programme gibt, die die Managementinformationen aus den SNMP-Nachrichten auslesen und grafisch oder statistisch direkt weiterverarbeiten können.

5.5 User Mode Linux Rechner


Ein UML-Rechner kann gut zum Testen von Linux-Software verwendet werden, da er nicht nur eine abgesicherte Linux-Betriebssystemumgebung bereitstellt, sondern auch die immer gleiche Linux-Betriebssystemumgebung, mit dem gleichen Ausgangszustand und den selben Einstellungen liefern kann.


Das Linux-Betriebssystem, das ein UML-Rechner liefert, ist ähnlich aufgebaut wie das eines reellen Rechners. Ein gewöhnliches Linux-Betriebssystem kann in zwei Komponenten unterteilt werden.

- **Einem Linux-Kernel**, der hardwareabstrahierenden Schicht des Betriebssystems in der die Ansteuerung der Geräte sowie die Speicher- und Prozessverwaltung stattfindet.

- **Einem Linux Root-Dateisystem**, in dem sich alle Anwendungsprogramme für den Benutzer befinden und auch die grundlegenden System-Programme, die das Betriebssystem erst funktionsfähig machen.

**Abb. 5.4: Aufbau eines Linux-Betriebssystems**


Um einen UML-Kernel zu erstellen muss allen kernspezifischen Befehlen die Option „ARCH=um“ angehängt werden. Vor allem die folgenden Befehle sind für die Erstellung eines UML-Kernels zu erwähnen, die innerhalb des Kernel-Quellcode-Verzeichnisses ausgeführt werden müssen:

- **make menuconfig ARCH=um** (alternativ.: make xconfig ARCH=um, make gconfig ARCH=um)
  Hiermit wird eine grafische Oberfläche zum konfigurieren des UML-Kernels gestartet. Die fertige Konfiguration wird dann in die Datei „.config“ innerhalb des Kernel-Verzeichnisses gespeichert.

- **make oldconfig ARCH=um**
  Ist eine Konfigurationsdatei („.config“) für die UML-Kernel Konfiguration bereits vorhanden, zum Beispiel legen die VNUML-Autoren ihren UML-Kernen eine Konfigurationsdatei bei, so können die Einstellungen mit diesem Befehl in den neuen Kernel übernommen werden.

- **make ARCH=um**

- **make modules_install INSTALL_MOD_PATH=’Root-Dateisystem-Mountpunkt’ ARCH=um**

  In dem Verzeichnis „/Root-Dateisystem-Mountpunkt“ ist hier die Imagedatei, die das Root-Dateisystem des UML-Rechners beinhaltet, im Host-System eingebunden.


### Listing 5.2: Inhalt der Datei „/etc/fstab“ des UML-Root-Dateisystems bei Verwendung von „devfs“


Mit den beiden UML-Rechner Komponenten der VNUML-Internetseite, dem UML-Kernel „linux-2.6.10-1m“ und dem Root-Dateisystem „root_fs_tutorial-0.2.3“ kann nun bereits mit dem folgenden Befehl ein einzelner UML-Rechner gestartet werden.

```
host # /linux-2.6.10-1m ubd0=uml_cow_fs,root_fs_tutorial-0.2.3 con0=null con=xterm
```

Befehl 5.9: Starten eines UML-Rechners


Die Imageadatei mit dem Root-Dateisystem des UML-Rechners, hier „root_fs_tutorial-0.2.3“, wird über die Befehlsoption ubd0 angegeben. Sie ist damit die erste Partition des UML Block-Devices (ubd).
Netzwerk-Simulationen mit Virtual Network User Mode Linux


UML bietet mit dem Tool „uml_moo“, das Bestandteil der „Usermode-Utilities“ ist, die Möglichkeit eine COW-Imagedatei mit seiner Basis-Imagedatei zu verschmelzen und so die Veränderungen im UML-Root-Dateisystem bleibend zu übernehmen.

```
host # uml_moo uml_cow_fs root_fs tutorial-0.2.3
```

Befehl 5.10: Verschmelzen einer Basis-Imagedatei mit der zugehörigen COW-Imagedatei

Mehrere UML-Rechner können zur gleichen Zeit gestartet werden und auf das gleiche Root-Dateisystem, das sich in der selben Basis-Imagedatei befindet, zugreifen. Jedem UML-Rechner wird so nur eine eigene COW-Imagedatei erstellt, in die er seine Daten ablegt. Da alle UML-Rechner nur lesend auf das original Root-Dateisystem in der Basis Imagedatei zugreifen, kann es hier auch nicht zu Konflikten zwischen den einzelnen UML-Rechnern kommen. Meist genügt ohnehin bei mehreren UML-Rechnern die zu einem Netzwerk zusammengeschlossen werden, wie VNUni das automatisch macht, ein einzelnes Root-Dateisystem, so dass alle UML-Rechner über die gleichen Anwendungsprogramme verfügen, die dann lediglich unterschiedlich konfiguriert werden müssen. Da eine Basis-Imagedatei mit einem Root-Dateisystem meist mehrere 100 Megabyte groß ist und die dazugehörige COW-Imagedatei im Vergleich dazu sehr klein ist, nur nach und nach größer wird und eigentlich nie ihre maximale Größe, die der Größe der Basis-Imagedatei entspricht, erreicht, spart der COW-Mechanismus auch eine Menge Speicherkapazität auf dem Host ein.


5 Netzwerk-Simulationen mit Virtual Network User Mode Linux


Mit dem Programm „brctl“ wird die Netzwerk-Bridge, hier mit Namen „Net0“, generiert. Das „Spanning Tree Protocol“ (STP) wird abgeschaltet da es standardmäßig aktiviert ist, hier aber nicht benötigt wird. Normalerweise dient STP der Vermeidung redundanter Pfade (Schleifen) im Netzwerk. Auch der Bridge wird die IP-Adresse 0.0.0.0 mit aktivem „promiscuous Modus“ zugeteilt.

tunctl -t uml1-eth0 -f /dev/net/tun
tunctl -t uml1-eth1 -f /dev/net/tun
ifconfig uml1-eth1 0.0.0.0 promisc up
tunctl -t uml2-eth0 -f /dev/net/tun
tunctl -t uml2-eth1 -f /dev/net/tun
ifconfig uml2-eth1 0.0.0.0 promisc up
brctl addbr Net0
brctlstp Net0 off
ifconfig Net0 0.0.0.0 promisc up

Befehl 5.11.1 : Aufbau des virtuellen Netzes im Host


ifconfig uml1-eth0 192.168.0.1 netmask 255.255.255.252 up
ifconfig uml2-eth0 192.168.0.5 netmask 255.255.255.252 up

Befehl 5.11.2 : Konfiguration der Management-Netzwerkschnittstellen des Hosts

Die Gegenstellen mit den IP-Adressen 192.168.0.2 und 192.168.0.6 müssen später, im entsprechenden UML-Rechner, nach dessen Start, eingerichtet werden. Die virtuellen Netzwerkschnittstellen der UML-Rechner, die miteinander in Verbindung stehen sollen und somit das virtuelle Netz aufspannen, müssen der Software-Bridge

```
brctl addif Net0 uml1-eth1
brctl addif Net0 uml2-eth1
```

Befehl 5.11.3: Hinzufügen der Netzwerkschnittstellen zur Bridge

Die virtuellen Netzwerkschnittstellen benötigen allerdings noch ihre IP-Adressen, die sie im virtuellen Netzwerk besitzen sollen. Diese müssen ebenfalls innerhalb der UML-Rechner gesetzt werden.


```
dd if=/dev/zero of=uml1-opt-fs count=1 bs=1M
mke2fs -F uml1-opt-fs
mkdir -o loop uml1-opt-fs mntuml1
mount -o loop uml1-opt-fs mntuml1
```

Befehl 5.11.4: Erstellen der Imagedatei für das Konfigurations-Script des UML-Rechners „uml1“


```
cat /root/.ssh/identity.pub >> mntuml1/authorized_keys
```

Befehl 5.11.5: Kopieren des SSH-Schlüssels


```
echo "#!/bin/bash" >> mntuml1/umlboot
echo "hostname uml1" >> mntuml1/umlboot
echo "/sbin/ifconfig eth0 192.168.0.2 netmask 255.255.255.252 up" >> mntuml1/umlboot
echo "/sbin/ifconfig eth1 0.0.0.0 promisc up" >> mntuml1/umlboot
echo "/sbin/ifconfig eth1 10.0.0.1 netmask 255.255.255.252 up" >> mntuml1/umlboot
echo "/sbin/route add default gw 10.0.0.2" >> mntuml1/umlboot
chmod a+x mntuml1/umlboot
umount mntuml1
```

Befehl 5.11.6: Erstellen der Script-Datei „umlboot“ für die Konfiguration des UML-Rechners „uml1“

Zum Schluss muss die Imagedatei aus dem Host-System gelöst werden, sonst kann es beim Start des UML-Rechners zu Problemen kommen.

Wie bereits erwähnt erstellt VNUML für jeden an der Netzwerk-Simulation beteiligten UML-Rechner eine solche Imagedatei mit individuell konfiguriertem Bash-Script „umlboot“ für die Konfiguration. Hier folgt die Konfiguration des UML-Rechners „uml2“.

```

dd if=/dev/zero of=uml2-opt-fs count=1 bs=1M
mke2fs -F uml2-opt-fs
mkdir mntuml2
mount -o loop uml2-opt-fs mntuml2

cat /root/.ssh/identity.pub >> mntuml2/authorized_keys

echo "#!/bin/bash" >> mntuml2/umlboot

echo "hostname uml2" >> mntuml2/umlboot

echo "/sbin/ifconfig eth0 192.168.0.6 netmask 255.255.255.252 up" >> mntuml2/umlboot

echo "/sbin/ifconfig eth1 0.0.0.0 promisc up" >> mntuml2/umlboot

chmod a+x mntuml2/umlboot

umount mntuml2
```

Befehl 5.11.7: Erstellen der Image- und Script-Datei für die Konfiguration des UML-Rechners „uml2“

Nun ist die Konfiguration beendet und die UML-Rechner können gestartet werden.

Ist beim Start einer Simulation mit VNUML die Befehlsoption für den „Blocking Mode“ (-B) angegeben, so wartet auch VNUML diese obige Konfiguration für alle beteiligten UML-Rechner ab, bevor sie dann parallel gestartet werden. Ist der „Blocking Mode“ nicht aktiviert, so wird jeder UML-Rechner einzeln konfiguriert und direkt im Anschluß daran gestartet.

Der Befehl zum Starten eines UML-Rechners wurde zuvor schon einmal im Befehl 5.9 gezeigt. Hier werden die beiden UML-Rechner mit den Imagedateien, die jeweils als erste und zweite Partition eingebunden werden, gestartet. Auch die Netzwerkschnittstellen, über die ein UML-Rechner verfügen soll, müssen beim Start des UML-Rechners angegeben werden. Dabei wird der verwendete Mechanismus (hier tuntap), die zugehörige Netzwerkschnittstelle im Host sowie die MAC-Adresse angegeben.


```

host # uml_mconsole /root/.uml/uml1/mconsole

(/root/.uml/) help
OK Commands:
   version - Get kernel version
   help - Print this message
   halt - Halt UML
reboot - Reboot UML
 [...]
```

Befehl 5.12 : Verwendung der UML-Management-Konsole „uml_mconsole“


Die Optionen „con“ und „con0“ konfigurieren die „Login-Connection“ zu den UML-Rechnern. Die Option „con=null“ schaltet den direkten Login über die Konsole aus. Die Option „con=xterm“ lässt ein X-Terminal


5.6 Hinweise zur Installation von Software in den UML-Rechner

Da sich die SNMP-Software Net-SNMP, die in dieser Studienarbeit verwendet wird, nicht so ohne weiteres in das Root-Dateisystem der VNUML-Autoren installieren lässt, wird im folgenden etwas ausführlicher auf den allgemeinen Vorgang der Installation neuer Software eingegangen.


Befehl 5.13: Vorbereitung der Imagedatei für die Software Installation


Mit dem Befehl „exit“ wird die „chroot“-Umgebung wieder verlassen und in das Host-System zurückgekehrt.

Befehl 5.14: Verlassen der „chroot“-Umgebung und lösen der Imagedatei


Die zweite Möglichkeit, Anwendungsprogramme in einen laufenden UML-Rechner zu installieren, zum Beispiel während einer VNUML-Simulation, wird im folgenden dargestellt.

Um einen einzelnen UML-Rechner im laufenden Betrieb an das Internet anzubinden, können die folgenden Schritte durchgeführt werden. Dabei wird auf den Host-Rechner ein NAT-Router (Network Address Translation) aktiviert, der den ein- und ausgehenden Datenverkehr des UML-Rechners durch das Host-System schleust. Wegen des NAT-Router befinde sich dann hinter der einen Internet-Adresse des Host-Systems zwei Rechner, zum einen der Host-Rechner selbst und zum anderen der UML-Rechner.


Die Debian APT-Tools verfügen, unter anderem, über die folgenden Optionen.

- **apt-get update**
  Synchronisiert die lokale Software-Distributionsliste der verfügbaren Softwarepakete mit einer globalen Distributionsliste im Internet. Der Befehl muss auch jedesmal ausgeführt werden, nachdem ein neuer Internet-Server in die Datei „sources.list“ eingetragen wurde, bevor von diesem Softwarepaket heruntergeladen und installiert werden können.

- **apt-cache search „Suchwort“**
  Findet alle Pakete auf der Software-Distributionsliste, deren Namen dem Suchwort ähneln oder mit dem Suchwort in einer Beziehung stehen.

- **apt-get install „Softwarepaket“**
  Lädt das angegebene Softwarepaket von einem Internet-Server herunter und installiert die enthaltene Software in das lokale System. Dabei wird auch überprüft, ob die Funktion der enthaltenen Software von weiteren Softwarepaketen abhängt, die dann ebenfalls heruntergeladen und installiert werden.

- **apt-get remove „Softwarepaket“**
  Deinstalliert die Software des angegebenen Paketes aus dem lokalen System.

- **apt-get source „Softwarepaket“**

- **apt-get build-dep „Softwarepaket“**
  Lädt die zu dem angegebenen Softwarepaket abhängigen Pakete aus dem Internet und installiert deren Software. Das ist zum Beispiel notwendig, wenn ein Programm aus dem Quellcode lokal kompiliert werden soll, dafür aber eine bestimmte Softwareumgebung für seine Funktionen benötigt.

Die Debian Installationsroutine installiert standardmäßig fertig kompilierte Binärpakete und überprüft die Abhängigkeiten der zu installierenden Softwarepakete zu anderen Paketen und installiert diese gegebenenfalls mit. Der Vorteil dabei ist, dass die gewünschte Software ohne großen Aufwand meist recht schnell und funktionsbereit zur Verfügung steht.

Ein Nachteil ist aber, dass die Software eben bereits konfiguriert und kompiliert ist und womöglich nicht gewollte Einstellungen übernommen werden müssen oder auf gewünschte Einstellungen verzichtet werden muss. Da das UML-Root-Dateisystem auch weitestgehend klein gehalten wird, kann es gerade hier passieren, dass die Installation eines Softwarepaketes über die Debian APT-Tools, die Installation einer großen Menge an zusätzlicher Software nach sich zieht.

Beim Debian werden zu jedem Zeitpunkt drei Versionen eines Softwarepaketes parallel angeboten, genannt stable (stabil), testing (prüfen) und unstable (instabil), die einem unterschiedlich fortgeschrittenen Software-Entwicklungszweig zugeordnet sind. Je nachdem aus welchem Zweig ein Softwarepaket installiert wird, müssen eventuell davon abhängige, bereits installierte Softwarepakete ebenfalls neu installiert werden. Es können aber nicht beliebig viele Anwendungsprogramme in das Root-Dateisystem installiert werden, da die Größe der Imagedatei den vorhandenen Speicherplatz beschränkt. Da manche Anwendungsprogramme jedoch, wie zum Beispiel die SNMP Software Net-SNMP, viel Speicherplatz benötigen, wird im folgenden nun kurz erläutert, wie die Imagedatei vergrößert werden kann.

<table>
<thead>
<tr>
<th>Befehl</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>host filesystems # e2fsck -f root_fs_tutorial-0.2.3</td>
<td></td>
</tr>
<tr>
<td>host filesystems # dd if=/dev/zero of=root_fs_tutorial-0.2.3 bs=1M count=1 seek=1000 conv=notrunc</td>
<td></td>
</tr>
<tr>
<td>host filesystems # resize2fs -p root_fs_tutorial-0.2.3</td>
<td></td>
</tr>
<tr>
<td>host filesystems # ls -lh root_fs_tutorial-0.2.3</td>
<td></td>
</tr>
<tr>
<td>552M -rwxr-xr-x 1 root root 1001M May 10 00:50 root_fs_tutorial-0.2.3</td>
<td></td>
</tr>
<tr>
<td>host filesystems # e2fsck -f root_fs_tutorial-0.2.3</td>
<td></td>
</tr>
</tbody>
</table>

Befehl 5.17: Vergrößern einer vorhanden Imagedatei


### 5.7 Die Routing-Software Quagga


Diese Multi-Prozess Architektur bringt Quagga eine gewisse Erweiterbarkeit, Modularität und ermöglicht eine einfache Wartung. Alle Quagga Routing-Dienste können über eine eigene Konfigurationsdatei, die beim Start...
angegeben werden muss, und während der Laufzeit, über eine „Telnet“-Schnittstelle, konfiguriert werden.


5.7.1 Installation


Die Software Quagga steht auf der Internetseite „http://www.quagga.net“ zum download bereit. Zwar kann sie im Root-Dateisystem der VNUML-Autoren auch als Binärpaket über die Online-Installationsroutine des Debian Linux-Systems installiert werden, allerdings unterstützt diese fertig kompilierte Version kein SNMP. Soll SNMP unterstützt werden muss die Installation manuell durchgeführt werden, was im folgenden gezeigt wird.

Allgemeine Hinweise zur Installation neuer Software innerhalb des UML-Root-Dateisystems der VNUML-Autoren sind im Kapitel 5.6 zu finden.

Der Quagga Quellcode kann auch mit Hilfe der bereits erwähnten Debian Bordmittel aus dem Internet automatisch heruntergeladen werden. Dafür muss in der Datei „/etc/apt/sources.list“ über das Schlüsselwort „deb-src“ eine Softwarequelltext-Quelle hinzugefügt wurde. Der folgende Befehl lädt dann die Quagga-Software automatisch aus dem Internet und entpackt sie im selben Verzeichnis.

```
/ # apt-get source quagga
```

*Befehl 5.19: Herunterladen und Entpacken des Quagga-Quellcodes*

Damit Quagga SNMP unterstützt und mit einem SNMP Agenten Daten über SMUX austauscht muss SNMP für das Kompilieren aktiviert werden. Um die Vorzüge des „vtysh“-Terminals zu nutzen und die laufenden Quagga Routing Dienste nicht einzeln via „Telnet“ ansteuern zu müssen, muss auch „vtysh“ aktiviert werden.


```
/ # ./configure --enable-snmp --enable-vtysh
/ # make && make install
```

*Befehl 5.20: Manuelle Installation von Quagga*


```
/ # which zebra
/ /usr/local/sbin/zebra
/ # which ripd
/ /usr/local/sbin/ripd
```

*Befehl 5.21: Lokalisation der Quagga Routing-Dienstprogramme*

Ist diese Referenz gesetzt, sollte die Verbindung via SMUX zwischen dem Quagga Routing-Dienstprogramm und Net-SNMP grundsätzlich möglich sein. Ist diese Referenz nicht gesetzt, so ist den einzelnen Routing Dienstprogrammen die Anweisung „smux peer“, welche die SMUX Verbindung aufsetzt, unbekannt.


```
/ # ldd /usr/local/sbin/zebra
[...]
libreadline.so.5 => /lib/libreadline.so.5 (0x400b3000)
lincurses.so.5 => /lib/libcurses.so.5 (0x400da000)
libnetsnmp.so.9 => /usr/lib/libnetsnmp.so.9 (0x40111000)
lbncrypt.so.0.9.7 => /usr/lib/libcrypt.so.0.9.7 (0x40195000)
[...]  
```

**Befehl 5.22: Überprüfen der Referenz auf die Net-SNMP-Bibliothek**

Diese MIB-Dateien müssen in das Standard-MIB-Verzeichnis der SNMP-Software kopiert werden. Bei Net-SNMP ist das für gewöhnlich „/usr/local/share/snmp/mibs/“ oder auch „/usr/share/snmp/mibs/“. Den Net-SNMP Befehlen muss noch der Zugriff auf die neuen MIB-Dateien erlaubt werden, was am einfachsten mit dem setzen der Umgebungsvariable MIBS="ALL" oder der stets angegebenen Befehlssoption -m ALL" erledigt werden kann. Ohne diese Einstellung kann nur über die numerische OID auf die Managementinformation zugegriffen werden.

```
/ quagga # find \ (-name "*MIB*" -or -name "*SMI*") \ -exec cp -vi \'/usr/local/share/snmp/mibs/ {}' \ /usr/local/share/snmp/mibs/ \;
./zebra/GNOME-SMI
// beinhaltet enterprises Objekte „gnome“ und „gnomeProducts“
./zebra/GNOME-PRODUCT-ZEBRA-MIB
// beinhaltet enterprises Objekte für die SMUX-Kommunikation
./bgpd/BGP4-MIB.txt
// beinhaltet die BGP4 Objekte { mib-2 15 }
./ripd/RIPv2-MIB.txt
// beinhaltet die RIPv2 Objekte { mib-2 23 }
./ospfd/OSPF-MIB.txt
// beinhaltet die OSPF Objekte { mib-2 14 }
./ospfd/OSPF-TRAP-MIB.txt
// beinhaltet die Objekte für Traps mit OSPF { ospf 16 }
```

**Befehl 5.23: Aufspüren der MIB-Dateien der Quagga Routing-Dienste**

Der Quagga Kernel-Routing-Table-Manager „zebra“ kann nach der erfolgreichen Installation wie folgt gestartet werden, wobei die Konfigurationsdatei „zebra.conf“, die mit der Befehlssoption -f beim Start angegeben wird, im Verzeichnis „/etc/quagga/“ abgelegt ist. Mit der Befehlssoption -d wird „zebra“ als Dienst gestartet.

```
# uml2 # zebra -f /etc/quagga/zebra.conf -d
```

**Befehl 5.24: Start des Quagga Kernel-Routing-Table-Managers „zebra“**

Jedes Quagga Routing-Dienstprogramm benötigt eine eigene Konfigurationsdatei, die beim Start des Dienstes angegeben werden muss. So wird der RIP-Routing Dienst „ripd“ wie folgt gestartet.

```
# uml2 # ripd -f /etc/quagga/ripd.conf -d
```

**Befehl 5.25: Start des Quagga RIP-Routers „ripd“**

Mit den anderen Routing-Dienstprogramm, die Quagga enthält, wird ebenso verfahren, wenn sie eingesetzt werden sollen. Alle Routing Dienste von Quagga können parallel ausgeführt werden und über „zebra“ auch Routen untereinander austauschen, so dass verschiedene Routing Protokolle gleichzeitig in einem System unterstützt werden.
5.7.2 Beispiele für Konfiguration und Umgang mit Quagga

Auf den Internetseiten der Quagga Autoren finden sich alle Konfigurationsmöglichkeiten für die einzelnen Quagga Routing-Dienste. Im folgenden werden Beispiele für die Konfiguration des Kernel-Routing-Table-Managers „zebra“ und des RIP-Routers „ripd“ aufgezeigt.


Die folgende Konfigurationsdatei für „zebra“ mit dem Namen „zebra.conf“, beinhaltet den Hostnamen, die Zugangspasswörter der „Telnet“-Schnittstelle zum Durchführen von Konfigurationen während der Laufzeit, die Angabe der Log-Datei und die Konfiguration der SMUX-Verbindung. Zeilen die mit einem „!“ (Ausrufezeichen) beginnen gelten als Kommentarzeilen. Der Eintrag „line vty“ aktiviert die Möglichkeit, den Routing Dienst auch über die „vtysh“-Terminal-Shell zu konfigurieren, was wesentlich komfortabler ist wenn mehrere Quagga Routing-Dienste auf einem Rechner konfiguriert werden müssen.


Die Konfiguration des Net-SNMP Agenten als SMUX-Gegenstelle und weitere Hinweise zum Abfragen der Managementinformationen sind im Kapitel 6.5.1.4 zu finden.


Wird der Routing-Dienst „zebra“ mit der oben aufgeführten Konfigurationsdatei gestartet, so kann während der Lauftzeit wie folgt via „Telnet“ in das Terminal des Dienstes eingeloggt werden.

```bash
hostname uml2zebra
password zebra
enable password enzebra
!
debug zebra events
line vty
log file /var/log/quagga/zebra.log
smux peer .1.3.6.1.4.1.3317.1.2.1 quagga_zebra
```


Wird der Routing-Dienst „zebra“ mit der oben aufgeführten Konfigurationsdatei gestartet, so kann während der Lauftzeit wie folgt via „Telnet“ in das Terminal des Dienstes eingeloggt werden.

```bash
um12 # telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost
Hello, this is Quagga (version 0.99.4). Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification
Password: zebra
um12zebra > show ip forwarding
IP forwarding is on
```

Statt beim Einloggen mit „Telnet“ die Portnummer anzugeben, hinter der die Terminal-Schnittstelle zu finden ist, wie hier im Beispiel dargestellt, kann auch der Name des Routing Dienstes angegeben werden, sofern dieser in der Datei „/etc/services“ mit der Portnummer verknüpft ist, was normalerweise der Fall sein sollte.
Soll die Routing-Konfiguration verändert werden, so muss in den „Enable Mode“ gewechselt werden.

<table>
<thead>
<tr>
<th>uml2zebra &gt; enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password: enzebra</td>
</tr>
<tr>
<td>uml2zebra#</td>
</tr>
</tbody>
</table>

Befehl 5.27: Wechseln in den „Enable Mode“ von „zebra“

In diesem Modus kann nach Eingabe der Anweisung „configure terminal“ die Konfiguration verändert werden, die dann mit der Anweisung „write“ auch bleibend in die beim Start angegebene Konfigurationsdatei übernommen werden kann. Alle gültigen Anweisungen, die innerhalb eines Terminals gelten, können über die Eingabe des Fragezeichens („?“) angezeigt werden.

<table>
<thead>
<tr>
<th>uml2zebra &gt; show ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>debugging Zebra configuration</td>
</tr>
<tr>
<td>history Display the session command history</td>
</tr>
<tr>
<td>interface Interface status and configuration</td>
</tr>
<tr>
<td>ip IP information</td>
</tr>
<tr>
<td>[...]</td>
</tr>
</tbody>
</table>

Befehl 5.28: Auflistung aller gültigen Anweisungen innerhalb „zebra“


<table>
<thead>
<tr>
<th>hostname uml2ripd</th>
</tr>
</thead>
<tbody>
<tr>
<td>password ripd</td>
</tr>
<tr>
<td>enable password enripd</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>line vty</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>router rip</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>network 10.0.0./24</td>
</tr>
<tr>
<td>network 164.1.2.0/24</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>smux peer .1.3.6.1.4.1.3317.1.2.3 quagga_ripd</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>log file /var/log/quagga/ripd.log</td>
</tr>
</tbody>
</table>

Listing 5.4: Inhalt der „ripd“-Konfigurationsdatei ripd.conf


<table>
<thead>
<tr>
<th>uml2 # telnet localhost 2602</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trying 127.0.0.1...</td>
</tr>
<tr>
<td>Connected to localhost</td>
</tr>
<tr>
<td>Hello, this is Quagga (version 0.99.4).</td>
</tr>
</tbody>
</table>

User Access Verification
Password: ripd

| uml2ripd > |

Befehl 5.29: Einloggen in die „ripd“-Terminal-Shell

Über den Befehl „vtysh“ kann in die Terminal-Shell eingeloggt werden.


Um die SMUX Verbindung für das Auslesen von Informationen via SNMP zu nutzen muss der SNMP-Dienst vor den Quagga Routing-Diensten gestartet werden und wie im Kapitel 6.5.1.4 beschrieben konfiguriert sein. In der Log-Datei des Net-SNMP Agenten „snmpd“ sollten sich dann, nach erfolgreichem zustande kommen der SMUX Verbindung mit „zebra“ und „ripd“, die folgenden Einträge finden.

Listing 5.5: Inhalt der Datei „snmpd.log“ nach Zustandekommen einer SMUX-Verbindung
Schlägt die Verbindung fehl, so wird dies in der Log-Datei des Net-SNMP Agenten und des entsprechenden Quagga Routing-Dienstes angezeigt.


5.8 Zusammenfassung


6 Net-SNMP in VNUML Simulationen

Um SNMP in einer VNUML-Netzwerk-Simulation verfügbar zu machen bedarf es der Installation einer SNMP-Software in das Root-Dateisystem, das in einem VNUML-Szenario für die UML-Rechner verwendet wird. In den gegenwärtigen Root-Dateisystemen, die die VNUML-Autoren auf ihre Internetseite zum Herunterladen bereitstellen, ist keine SNMP-Software installiert.

In diesem Kapitel wird die SNMP Implementierung Net-SNMP vorgestellt, installiert und Möglichkeiten der Konfiguration behandelt. Einige der Manager- und Agentenfunktionen, die Net-SNMP bietet, werden vorgestellt und in Beispielen angewendet.

6.1 Einleitung


Bei Net-SNMP handelt es sich um ein Softwarepaket, das aus mehreren Anwendungsprogrammen besteht und eine vollständige SNMP-Umgebung, also Manager und Agenten, aufbauen kann. Unterstützt werden alle derzeitigen SNMP Versionen, also SNMPv1, -v2c und -v3.

Insgesamt beinhaltet das Net-SNMP Softwarepaket folgende Programme:

- Eine Anzahl von Kommandozeilen-Programmen mit denen SNMP-Befehle und -Meldungen gesendet werden können.
- Darauf aufbauende „Second-Level“-Kommandozeilen-Programme, die den Umgang mit SNMP für spezielle Aufgaben erleichtern.
- Einen Agenten- sowie einen Trap-Server mit denen SNMP-Befehle, bzw. -Meldungen, empfangen, verarbeiten und beantwortet werden können.
- Eine grafische Benutzeroberfläche für den Manager
- Eine Programm-Bibliothek sowie verschiedene „Helfertools“, die zur Erweiterung und Konfiguration des Net-SNMP Agenten eingesetzt werden können.


6.2 Entwicklung und geschichtlicher Hintergrund zu Net-SNMP


Wes Hardaker, System Administrator an der University of California Davis (UCD), installierte das CMU-SNMP Paket und erweiterte den enthaltenen Agenten um mehr Informationen über die von ihm verwalteten Systeme zu erhalten und um Fehlerzustände besser erkennen zu können. Dabei vereinfachte er die Möglichkeit neue Managementinformationen in den Agenten einzubinden. Das Ergebnis war ein einfach zu erweiternder


Laut Net-SNMP Projektgruppe liegt das Projekt in der Sourceforge Statistik der „Most Active Projects“, seit seiner dortigen Anmeldung, in den oberen 2%, was auch andeutet wie belebt dieses Software-Projekt ist.

### 6.3 Installation von Net-SNMP


Zwar gibt es über die Debian-Installationsroutine die Möglichkeit den Net-SNMP Agenten getrennt vom Net-SNMP Manager zu installieren, bei der manuellen Installation der Software werden aber standardmäßig alle notwendigen Anwendungsprogramme des Net-SNMP Softwarepakettes installiert.


Hier wird also die manuelle Installation des Net-SNMP Softwarepaketes empfohlen, von der für diese Studienarbeit die Version 5.2.2 gewählt wurde. Die derzeit schon verfügbare, neuere Version 5.3.1 funktionierte zwar ebenfalls einwandfrei, allerdings war es Quagga während der praktischen Ausarbeitung dieser Studienarbeit nicht möglich auf die SNMP-Bibliotheken dieser Version für den erfolgreichen Aufbau einer SMUX Verbindung zuzugreifen.

Allgemeine Hinweise zur Installation von Programmen in die Imagedatei des VNUML-Root-Dateisystems werden in Kapitel 5.6 aufgeführt.


deb-src ftp://ftp.debian.org/debian stable main contrib non-free

Listing 6.1: Eintrag einer Quelltext-Downloadquelle für die Debian APT-Tools in die Datei „sources.list“

```
/ # apt-get update
/ # apt-get source net-snmp
```

Befehl 6.1: Download des Net-SNMP Quellcodes mittels Debi ans „apt-get“


```
/ # apt-get build-dep net-snmp
```

Befehl 6.2: Installation der Software, die Net-SNMP für seine Funktion benötigt


Nachdem entpacken des Quellcodes kann Net-SNMP beispielsweise wie folgt für das Kompilieren und die anschließende Installation vorbereitet werden. (Die Backslashes am Ende der Zeilen zeigen lediglich den Zeilensprung an.)

```
/ net-snmp # /configure --with-openssl=/usr/bin(openssl) --enable-ucd-snmp-compatibility
   --enable-embedded-perl --with-perl-modules --enable-shared
   --with-mib-modules="smux disman/event-mib host ucd-snmp/diskio"
   --enable-mdf-rewrites
```

Befehl 6.3: Konfiguration für das Kompilieren von Net-SNMP


Mit dem folgenden Befehl wird Net-SNMP schließlich kompiliert und installiert.

```
/net-snmp # make && make install
```

Befehl 6.4: Installation von Net-SNMP

Nach der Installation sind sämtliche Net-SNMP Programme global in der Konsole ausführbar. Zusätzliche Dateien, wie die MIB-Dateien in denen die SMI-Objekt-Definitionen der Managementinformationen zu finden sind, befinden sich standardmäßig im Verzeichnis „/usr/local/share/snmp“ oder auch im Verzeichnis „/usr/share/snmp“.

An dieser Stelle sei noch angemerkt, dass während der praktischen Ausarbeitung der Studienarbeit festgestellt wurde, dass der Net-SNMP Agent die Managementinformationen seines zu verwaltenden Systems in einem Cache zwischenspeichert und diesen stellenweise nur minütlich aktualisiert. Dies hat im normalen Einsatz den Vorteil, dass der Agent schnell und direkt auf Managementinformationen zugreifen kann ohne das zu verwaltende System aufgrund der Anfrage eines oder mehrere Manager übermäßig zu belasten. Für eine Simulations-Umgebung ist es jedoch nicht unbedingt wünschenswert, wenn zum Beispiel das Zähler-Objekt des ein- und ausgehende Datenverkehrs der Netzwerkschnittstellen (if\[In/Out\]Octets) nur einmal pro Minute aktualisiert wird und der Manager sich so auch nur minütlich über den Verkehrsfluss der Netzwerkschnittstelle informieren kann. Der Versuch den Cache ganz abzuschalten ließ den Agenten überhaupt nicht mehr auf SNMP-Anfragen reagieren. Letztlich konnte zumindest für das Beispiel der Netzwerkschnittstellen-Aktualisierung Abhilfe geschaffen werden in dem, vor dem Komplieren des Quellcodes, die Variable „IFTABLE_CACHE_TIMEOUT“ aus der Datei „ifTable_data_access.h“, die sich im Verzeichnis „agent/mibgroup/if-mib/ifTable“ innerhalb des Net-SNMP Quelltext-Verzeichnisses befindet, auf einen kleineren Wert, zum Beispiel 1, gesetzt wurde. Somit aktualisiert der Agent die Managementinformationen der Objekte in der Interface-Tabelle, die die Verkehrsdaten der Netzwerkschnittstellen verwaltet, jede Sekunde.

### 6.4 Der Net-SNMP Manager


#### 6.4.1 Konfiguration des Net-SNMP Managers

Die Konfiguration des Net-SNMP Managers ist auf zwei Wegen möglich. Der erste Weg ist eine allgemeingültige Konfiguration über eine Konfigurationsdatei, der zweite Weg eine spezifische Konfiguration, die bei jeder Ausführung eines Kommandozeilen-Programms über entsprechende Befehlsoptionen des Programms mit angegeben werden kann. Die grundlegende Konfiguration zur Funktion des Managers, wie zum


Die Konfigurationsmöglichkeiten des Net-SNMP Managers lassen sich in vier Kategorien einteilen:

- **Textuelles MIB-Parsen**
  Diese Kategorie bietet Konfigurationsmöglichkeiten die den Umgang mit der MIB des Managers definieren, zum Beispiel in welchem Verzeichnis die MIB-Dateien abgelegt sind, welche Management-Objekte der MIB eingebunden sein sollen, usw.

- **Standard-Authentifizierungs Optionen**

- **Debug Ausgabe**
  Hier wird konfiguriert wie ausführlich die Ausgabe sein soll, um zum Beispiel Konfigurationsfehler besser finden zu können.

- **AusgabefORMAT Optionen**
  Bietet Konfigurationsmöglichkeiten bezüglich des Ausgabeformats der eintreffenden SNMP-Nachrichten. Zum Beispiel kann hier eingestellt werden, dass eintreffende Managementinformationen so ausgegeben werden, dass sie einfach in Variablen anderer Programme gespeichert und direkt weiterverarbeitet werden können.


**6.4.2 Anwendungsprogramme des Net-SNMP Managers**

Der Manager von Net-SNMP verfügt über mehrere Kommandozeilen-Programme, welche die verschiedenen SNMP-Befehle generieren, an den Agenten senden, dessen Antwort empfangen und diese auf der Konsole ausgeben.

Net-SNMP baut auf einem einheitlichen Kommandosatz auf, so dass sich alle Kommandozeilen-Programme in ihrer Verwendung ähneln. Die Kommandozeilen-Programme sind im allgemeinen wie folgt aufgebaut.

```
snmpcmd [Optionen] Agent [Parameter]
```

**Befehl 6.5: Allgemeines Verwendungs-Schema der Kommandozeilen-Programme des Managers**

Für die anzusteuernden Objekte muss nicht unbedingt deren nummerische OID im Parameterfeld angegeben werden. Die Objekte können auch über ihre Namen angesteuert werden, sofern der Net-SNMP Manager Zugriff auf die SMI-Definition des Objektes hat, aus der die Manager Programme die nummerische OID für das versenden letztlich ablesen. So kann beispielsweise die Systembeschreibung eines Agenten, die über das Management-Objekt „sysDescr“ aus der Gruppe „system“ verwaltet wird, wie folgt angesteuert werden.

Listing 6.2: Möglichkeiten der Objekt-Bezeichnung des Managers

Es können auch Mischformen zwischen nummerischen und textuellen OIDs verwendet werden. Damit die textuellen Objektnamen verwendet werden können, muss der Net-SNMP Manager Zugriff auf die textuelle „Management Information Base“ haben. Dafür muss den Net-SNMP Programmen das Verzeichnis bekannt sein, in dem die MIB-Dateien abgelegt sind. Standardmäßig benutzen die Net-SNMP Kommandoziele-Programme das Verzeichnis „/usr/local/share/snmp/mibs/“. Neue MIB-Dateien, deren MIB-Module nicht mit Net-SNMP kompiliert und installiert wurden, müssen in diesem Verzeichnis abgelegt werden und sie müssen meist auch über die Befehlsoption (-m) oder über die Umgebungvariable MIBS eingebunden sein. Mit dem Befehl „export MIBS="ALL"“ wird die Umgebungvariable MIBS so gesetzt, dass alle MIB-Dateien, die sich im Standard-MIB-Verzeichnis befinden, von Net-SNMP verwendet werden. Erlaubt aber ein Agent grundsätzlich den Zugriff auf eine Managementinformation, so sollte diese vom Manager aus immer mit der nummerischen OID angesteuert werden können und über die textuelle OID immer nur dann, wenn der Manager zusätzlich auf die MIB-Datei, in der sich die SMI-Definition des Objektes befindet, zugreifen kann. Ob dies der Fall ist kann mit dem Kommandozeilen-Programm „snmptranslate“ festgestellt werden.

Im folgenden werden nun die wichtigsten Net-SNMP Kommandozeilen-Programme des Managers mit je einem Beispiel für ihre Benutzung erläutert. In den Beispielen wird immer die niedrigste SNMP Version verwendet, die für das Kommandozeilen-Programm funktioniert. Dies schließt die Funktion für die darüber liegenden Versionen mit ein. Das Beispiel basiert auf der Netztopologie der isolierten Simulations-Umgebung aus Abbildung 6.1. Der Manager besitzt die IP-Adresse 10.0.0.1 und der Agent die IP-Adresse 10.0.0.2.


Der Net-SNMP Manager besitzt die folgenden Programme für die Ausführung von SNMP-Befehlen:

- **snmpget:**

Manager # snmpget -v 1 -c public 10.0.0.2 SNMPv2-MIB::sysDescr.0 SNMPv2-MIB::sysContact.0
SNMPv2-MIB::sysDescr.0 = STRING: Linux Agent 2.6.12.6-1m #5 Mon Jul 3 23:32:43 UTC 2006 i686
SNMPv2-MIB::sysContact.0 = STRING: root@Agent

Befehl 6.6: Get-Anfrage mit Net-SNMPs „snmpget“
Das gleiche Beispiel nochmal ausgeführt, diesmal mit SNMPv3:

Manager # snmpget -v 3 -u myuser -l authPriv -a MD5 -A auth_password -x DES -X priv_password 10.0.0.2 SNMPv2-MIB::sysDescr.0 SNMPv2-MIB::sysContact.0

SNMPv2-MIB::sysDescr.0 = STRING: Linux Agent 2.6.12.6-1m #5 Mon Jul 3 23:32:43 UTC 2006 i686
SNMPv2-MIB::sysContact.0 = STRING: root@Agent

Befehl 6.7: SNMPv3 Get-Anfrage mit Net-SNMPs „snmpget“

**snmpgetnext:**

Entspricht dem GetNext-Befehl der SNMP-Spezifikation. Die Managementinformation des Objektes, dessen OID die nächst Größere gegenüber der des angegebenen Objektes ist, wird ausgelesen. Im Beispiel wird die Systembeschreibung, deren OID wegen fehlender Angabe der Instanz folgt, und der Systemname, dessen OID auf die der Kontaktinformation folgt, ausgelesen.

Manager # snmpgetnext -v 1 -c public 10.0.0.2 SNMPv2-MIB::sysDescr SNMPv2-MIB::sysContact.0

SNMPv2-MIB::sysDescr.0 = STRING: Linux Agent 2.6.12.6-1m #5 Mon Jul 3 23:32:43 UTC 2006 i686
SNMPv2-MIB::sysName.0 = STRING: Agent

Befehl 6.8: GetNext-Anfrage mit Net-SNMPs „snmpgetnext“

**snmpbulkget:**

Entspricht dem GetBulk-Befehl der SNMPv2-Spezifikation. Über die zusätzlich angegebenen Parameter „non-repeaters“ (-Cn) und „max-repetitions“ (-Cr) werden die auszulesenden Objekte eingegrenzt. Der erste Wert gibt die Anzahl der ersten OIDs auf der abzufragenden OID-Liste an, auf die nur eine einzige GetNext-Operation angewendet werden soll, der zweite Wert gibt die Anzahl der GetNext-Operationen an, die auf die übrigen OIDs auf der abzufragenden OID-Liste angewendet werden sollen. Im Beispiel wird, wegen fehlender Angabe von Instanz, der Systemname ausgelesen und die jeweils nächsten drei Werte unterhalb der OID der Netzwerkschnittstellen-Beschreibung „ifDescr“ und unterhalb der OID des Objektes „ifInOctets“, das die Anzahl der eingehenden Bytes der Netzwerkschnittstelle definiert.

Manager # snmpbulkget -v 2c -c public -Cn1 -Cr3 10.0.0.2 SNMPv2-MIB::sysName IF-MIB::ifDescr IF-MIB::ifInOctets

SNMPv2-MIB::sysName.0 = STRING: Agent
IF-MIB::ifDescr.1 = STRING: eth1
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: lo
IF-MIB::ifInOctets.1 = Counter32: 13443
IF-MIB::ifInOctets.2 = Counter32: 17709
IF-MIB::ifInOctets.3 = Counter32: 0

Befehl 6.9: GetBulk-Anfrage mit Net-SNMPs „snmpbulkget“

**snmpset:**


Manager # snmpset -v1 -c private 10.0.0.2 SNMPv2-MIB::sysContact.0 s "admin@agent"

SNMPv2-MIB::sysContact.0 = STRING: admin@agent

Befehl 6.10: Set-Aufforderung mit Net-SNMPs „snmpset“
• *snmpwalk, snmpbulkwalk:*


Manager # snmpwalk -v1 -c public 10.0.0.2 udp

UDP-MIB::udpInDatagrams.0 = Counter32: 146
UDP-MIB::udpNoPorts.0 = Counter32: 12
UDP-MIB::udpInErrors.0 = Counter32: 0
UDP-MIB::udpOutDatagrams.0 = Counter32: 133
UDP-MIB::udpLocalAddress.0.0.0.0.161 = IpAddress: 0.0.0.0
UDP-MIB::udpLocalPort.0.0.0.0.161 = INTEGER: 161

Befehl 6.11: Auflistung aller verfügbaren Managementinformationen der UDP-MIB mit „snmpwalk“

• *snmptranslate:*

Dieses Kommandozeilen-Programm entspricht keiner SNMP Spezifikation. Mit diesem Programm kann der komplett verfügbare ISO-Registrierungsbaum und damit die „Management Information Base“ des Managers nach einem Objekt durchsucht und die Definition des Objektes ausgelesen werden. Auch die Baumstruktur, die unter dem angegebenen Objekt aufgebaut ist, kann mit diesem Programm angezeigt werden. Im Beispiel links wird die Baumstruktur der UDP-MIB angezeigt und im Beispiel rechts die SMI-Definition des Objektes „udpLocalAddress“ aus der UDP-MIB.

Manager # snmptranslate -Tp -IR udp
   +--udp(7)
   |    +--R- Counter udpInDatagrams(1)
   |    +--R- Counter udpNoPorts(2)
   |    +--R- Counter udpInErrors(3)
   |    +--R- Counter udpOutDatagrams(4)
   |    +--udpTable(5)
   |        +--udpEntry(1)
   |            Index: udpLocalAddress, udpLocalPort
   |            +--R- IpAddr udpLocalAddress(1)
   |            +--R- INTEGER udpLocalPort(2)
   |            Range: 0..65535

Manager # snmptranslate -Td -IR UDP-MIB::udpLocalAddress
UDP-MIB::udpLocalAddress udpLocalAddress OBJECT-TYPE
   -- FROM UDP-MIB, RFC1213-MIB
   SYNTAX IpAddress
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION "The local IP address for this UDP listener. In the case of a UDP listener which is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used.

::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) udp(7)
udpTable(5) udpEntry(1) 1 }"

Befehl 6.12: Abfrage von Objekt-Definitionen mittels „snmptranslate“


Dies sind die wichtigsten Net-SNMP Kommandozeilen-Programme des Managers.
Net-SNMP in VNUML Simulationen

Der Vorteil der Kommandozeilen-Programme des Net-SNMP Managers ist die Möglichkeit, sie in größere Script-Programme, zum Beispiel Bash- oder Perl-Programme, einzubetten. So können auf schnellem und einfachem Weg umfangreichere und komplexere SNMP-Anfragen mit beliebigem Ausgabeformat erstellt werden.

Darüber hinaus verfügt der Net-SNMP Manager noch über einige weitere „Second-Level“-Kommandozeilen-Programme, die den Umgang mit SNMP-Befehlen für bestimmte Aufgaben vereinfachen. So eignet sich das Kommandozeilen-Programm „snmptable“ sehr gut für die Ausgabe von Tabellen, da es die Managementinformation in einer übersichtlichere Tabellenstruktur auf der Konsole ausgibt und nicht einfach nur untereinander auflistet. Das Programm „snmptable“ verwendet unter SNMPv1 das Programm „snmpwalk“ und ab SNMPv2c das Programm „snmpbulkwalk“.


Mit dem Programm „snmpd“ werden dem Manager via SNMP alle eingehängten Partitionen des Agentensystems angezeigt, sowie die gesamte, belegte und freie Speicherkapazität der Partitionen.

Das folgende Bash-Programm ist ein Beispiel für die Einbettung von Net-SNMP Kommandozeilen-Programmen des Managers. Es bekommt als Übergabeparameter die IP-Adresse des Agenten sowie den dort gültigen Community-Namen und fragt über mehrere SNMPv1-Befehle die Identifikation und den bisherigen Ein- und Ausgangsverkehr des Agenten ab. Im ersten Abschnitt des Programms wird der Name, die Beschreibung, der Kontakt und die Anzahl der Netzwerkschnittstellen vom Agenten jeweils mit einem SNMPv1-Befehl abgefragt. Im zweiten Teil des Programms läuft dann eine While-Schleife über die zuvor ermittelte Anzahl der Netzwerkschnittstellen und fragt von diesen die Menge der bisher ein- und ausgegangenen Bytes ab.

```bash
#!/bin/bash
AGENT_IP=$1
COM=$2
AGENT_NAME=`snmpget -Oqv -v1 -c $COM $AGENT_IP system.sysName.0`
AGENT_DESCRIPTION=`snmpget -Oqv -v1 -c $COM $AGENT_IP system.sysDescr.0`
AGENT_CONTACT=`snmpget -Oqv -v1 -c $COM $AGENT_IP system.sysContact.0`
IFNUMBER=`snmpget -Oqv -v1 -c $COM $AGENT_IP interfaces.ifNumber.0`

echo "Agent: $AGENT_NAME   Contact: $AGENT_CONTACT"
echo "$AGENT_DESCRIPTION"

set -- $IFNUMBER
ifnum=$1
count=0
echo "Interfaces: IN       OUT"

while [ "$count" -lt "$ifnum" ]
do
count=$((count+1))
IFDESCR=`snmpget -Oqv -v1 -c $COM $AGENT_IP ifDescr.$count`
IFINOCTS=`snmpget -Oqv -v1 -c $COM $AGENT_IP ifInOctets.$count`
IFOUTOCTS=`snmpget -Oqv -v1 -c $COM $AGENT_IP ifOutOctets.$count`
echo "$count: $IFDESCR  : $IFINOCTS Byte, $IFOUTOCTS Byte"
done
exit 0
```

Listing 6.3: Bash-Script „snmpQuery.sh“ mit eingebundenen Net-SNMP Anwendungen des Managers
Wird dieses Bash-Programm in eine Datei mit Namen „snmpQuery.sh“ abgelegt und werden die Ausführungsrechte entsprechend gesetzt (chmod 755 snmpQuery.sh), so kann es wie folgt ausgeführt werden.

```bash
Manager # ./snmpQuery.sh 10.0.0.2 public
AGENT: Agent Contact: admin@agent
Linux Agent 2.6.12.6-1m #5 Mon Jul 3 23:32:43 UTC 2006 i686
Interfaces: IN OUT
1: eth1 : 55138 Byte, 69862 Byte
2: eth0 : 102991 Byte, 103895 Byte
3: lo : 2016 Byte, 2016 Byte
```

Die Abfrage ist an den SNMP-Agenten mit der IP-Adresse 10.0.0.2 und dem Community-Namen public gerichtet und gibt in dieser Reihenfolge Name, Kontaktinformation, System-Beschreibung und die bisher eing- und ausgegangenen Bytes der einzelnen Netzwerkschnittstellen des Agenten aus.

Da hier jedes Management-Objekt über einen eigenen SNMP-Befehl abgefragt wird, ist der Datenverkehr, der von diesem Programm erzeugt wird, relativ hoch. Das Programm kann bezüglich der Menge der zu übertragenen Daten erheblich optimiert werden, wenn über einen SNMP-Befehl mehrere Objekte abgefragt werden.

### 6.4.3 Der Net-SNMP Trap-Server


Mit dem Befehl „snmptrapd --help“ wird eine Liste von möglichen Startkonfigurationen angezeigt.

Der folgende Befehl startet den Trap-Server und gibt die eingehenden Meldungen direkt auf der Konsole aus.

```bash
Manager # snmptrapd -f
```

Standardmäßig lauscht der Net-SNMP Trap-Server auf dem Netzwerkport 162 auf eingehende SNMP-Meldungen.


```bash
Manager # snmptrapd -c /etc/snmp/snmptrapd.conf -f /var/log/snmptrapd.log
```


Mit dem Perl-Programm „snmpconf“, das im Net-SNMP Softwarepaket enthalten ist, kann über eine übersichtliche Menüsteuerung ebenfalls die Konfigurationsdatei für den Trap-Server erstellt werden.

In der Konfigurationsdatei können Zugriffsnamen, Community- oder SNMPv3-Benutzernamen, eingestellt werden, so dass nur die Meldungen vom Trap-Server verarbeitet werden, die einen passenden Zugriffsnamen besitzen. Die Net-SNMP Version 5.2.2 ist bei der Behandlung von SNMPv1 und -v2c Meldungen etwas abweichend zur Version 5.3.1. Bei der muss eine „authCommunity“ mit einer Zuteilung der Zugriffsrechte „log“ (Protokollieren), „execute“ (Auszählen) oder „net“ (Weiterleiten) angegeben werden und nur die

```
## ACCESS CONTROL ##
#SNMPv3
createUser mytrapuser MD5 "md5_password" DES "des_password"
createUser trUser MD5 "trap_password" DES "other_password"

#SNMPv1, -v2c
trapcommunity trapCom
trapcommunity trapatonie
trapcommunity publictrap
```

**Listing 6.4.1: Konfiguration der Zugriffsrechte in der Datei „snmptrapd.conf“ des Trap-Server**

Um externe Programme nach dem Erhalt einer Meldung starten zu können muss die Anweisung „traphandle“ und die OID des Objektes, das die Meldung verursacht, zusammen mit dem auszuführenden Programm verknüpft werden. Im angegebenen Beispiel in Listing 6.4.2 werden durch empfangene Meldungen, ausgelöst durch unterschiedliche Objekte, verschiedene Programme oder die gleichen Programme mit unterschiedlichen Parametern gestartet.

```
## TRAP HANDLE ##
# OID                  ausf. Anwendung(opt) Scriptname Argumente
traphandle SNMPv2-MIB::coldStart  /tmp/trapProg.sh start
traphandle NET-SNMP-AGENT-MIB::nsNotifyShutdown /tmp/trapProg.sh shutdown
traphandle IF-MIB::linkDown       /tmp/ifTrapProg.sh down
traphandle IF-MIB::linkUp         /tmp/ifTrapProg.sh up
traphandle .1.3.6.1.4.1.2021.2.1.100 /tmp/progieControl.sh Error
traphandle default                /bin/bash       /tmp/trapReceiver
```

**Listing 6.4.2: Konfiguration des Trap-Servers für das Verarbeiten von Meldungen**


6.5 Der Net-SNMP Agent


6.5.1 Konfiguration des Net-SNMP Agenten


<table>
<thead>
<tr>
<th>Agent #</th>
<th>snmpconf -g basic_setup</th>
</tr>
</thead>
</table>

Befehl 6.16: Erstellen einer Konfigurationsdatei für den Agenten mittels „snmpconf“


<table>
<thead>
<tr>
<th>Agent #</th>
<th>snmpd -c /etc/snmp/snmpd.conf -Lf /var/log/snmpd.log</th>
</tr>
</thead>
</table>

Befehl 6.17: Starten des Net-SNMP Agenten

Sämtliche Startoptionen des Net-SNMP Agenten werden mit dem Befehl „snmpd --help“ aufgezeigt

Es gibt eine sehr große Anzahl an Konfigurationsanweisungen für den Net-SNMP Agenten, die sich in die folgenden vier Kategorien einteilen lassen.

Es gibt Konfigurationsanweisungen die:

- SNMPv1/-v2c Community-Namen und SNMPv3-Benutzernamen sowie deren Zugriffsrechte konfigurieren.
- bestimmte Objekte, bzw. deren Managementinformationen, des Agenten konfigurieren können.
- die Überwachung von Management-Objekten konfigurieren und festlegen, wann und wegen welchem Objekt an welchen Manager Meldungen gesendet werden.
- Erweiterungen des Agenten konfigurieren, zum Beispiel den Anschluß an Sub-Agenten.


In den folgenden Kapiteln werden nun einige grundlegende Konfigurationsanweisungen über Beispiele dargestellt.

6.5.1.1 Konfiguration von Zugriffsrechten

Die schnellste und einfachste Möglichkeit Zugriff auf die Objekte und deren Managementinformationen des Net-SNMP Agenten zu bekommen, ist die Konfiguration von SNMPv1/-v2c Zugriffsmöglichkeiten durch die Erstellung von „read-write“ (RW) oder „read-only“ (RO) Communities. Im folgenden Beispiel wird die RW-Community „public“ und die RO-Community „private“ erstellt, die keinerlei weitere Einschränkungen besitzen,
sowie die RO-Community „publicSys“, die lediglich Lesezugriff auf die Objekte der Gruppe „system“ der MIB-II besitzt und deren Anfragen nur aus dem IP-Bereich des 10.0.0.0/24er Netzes eingehen dürfen.

<table>
<thead>
<tr>
<th>#</th>
<th>Community</th>
<th>NET</th>
<th>MIB-Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>rocommunity</td>
<td>publicSys</td>
<td>10.0.0.0/24</td>
<td>system</td>
</tr>
</tbody>
</table>

Listing 6.5.1: Konfiguration für den SNMPv1 / -v2c Zugriff in der Datei „snmpd.conf“


Listing 6.5.2: Konfiguration für den SNMPv3 Benutzerzugriff in der Datei snmpd.conf


Um VACM zu konfigurieren nutzt Net-SNMP vier Schlüsselwörter,
- • com2sec ordnet Community-Namen bestimmte Security-Namen zu (gilt nur für IPv4-Netze)
- • group Fasst die Security-Namen mit Sicherheitsmodellen zu Gruppen zusammen
- • view Erstellt Sichten über den ISO-Registrierungsbaum, bzw. die MIB
- • access Regelt den Zugriff und verknüpft die erstellten Gruppen mit den Zugriffsberechtigungen und -sichten

Im ersten Abschnitt werden die Community-Namen und sogenannte Security-Namen für Zugriffe aus verschiedenen IPv4-Netzbereichen festgelegt. Für Zugriffe über Verbindungen anderer Art gibt es andere Anweisungen, zum Beispiel für Zugriffe über Unix-Sockets die Anweisung com2secunix.

<table>
<thead>
<tr>
<th># Access Control, define security names</th>
</tr>
</thead>
<tbody>
<tr>
<td># sec.-name source community</td>
</tr>
<tr>
<td>com2sec manager localhost myprivate</td>
</tr>
<tr>
<td>com2sec manager 10.0.0.1 admin</td>
</tr>
<tr>
<td>com2sec network 10.0.0.0/24 mynetwork</td>
</tr>
<tr>
<td>com2sec world default allpublic</td>
</tr>
<tr>
<td>createUser InterfaceAdmin MD5 &quot;ifmd5_password&quot; DES &quot;ifdes_password&quot;</td>
</tr>
</tbody>
</table>

Hier wird der Vollständigkeit wegen auch gleich der SNMPv3-Benutzer „InterfaceAdmin“ erstellt.

Im zweiten Abschnitt werden die zuvor definierten Security-Namen mit Sicherheitsmodellen zu Gruppen zusammengefasst. Die Security-Namen entsprechen übrigens auch SNMPv3-Benutzernamen, das bedeutet, dass ab hier auch SNMPv3-Benutzer in das VACM-Modell eingebunden werden können.

<table>
<thead>
<tr>
<th># Map security names to groups</th>
</tr>
</thead>
<tbody>
<tr>
<td># groupname sec.-model sec.-name</td>
</tr>
<tr>
<td>group untrusted v1 world</td>
</tr>
<tr>
<td>group R0group v1 network</td>
</tr>
<tr>
<td>group R0group v2c network</td>
</tr>
<tr>
<td>group RWgroup v1 manager</td>
</tr>
<tr>
<td>group RWgroup v2c manager</td>
</tr>
<tr>
<td>group IFgroup usm InterfacesAdmin</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th># create a view to let the groups have the rights to</th>
</tr>
</thead>
<tbody>
<tr>
<td># name incl/excl subtree</td>
</tr>
<tr>
<td>view all included .1</td>
</tr>
<tr>
<td>view descr included system.sysDescr</td>
</tr>
<tr>
<td>view mib-2 included iso.org.dod.internet.mgmt.mib-2</td>
</tr>
<tr>
<td>view ifview included interfaces</td>
</tr>
</tbody>
</table>

Listing 6.5.3: Konfiguration von VACM Security-Namen in der Datei „snmpd.conf“

Listing 6.5.4: Konfiguration von VACM Gruppen in der Datei „snmpd.conf“

Listing 6.5.5: Konfiguration von VACM-Sichten in der Datei „snmpd.conf“
Im vierten und letzten Abschnitt schließlich wird festgelegt welche Gruppen mit welchen Zugriffsrechten auf welche Sichten zugreifen dürfen.

Besitzt ein Agent diese Konfiguration, so bekommt zum Beispiel ein Mitglied der „allpublic“-Community auf die folgenden Anfragen an den Agenten die folgende Ausgaben auf dem Manager.

Listing 6.5.6: Konfiguration von VACM Access Policies in der Datei „snmpd.conf“

Listing 6.5.7: Konfiguration von Systeminformationen des Agenten in der Datei „snmpd.conf“

Die „allpublic“-Community erhält vom Agenten keine Antwort auf SNMPv2-Befehle. Sie erhält nur auf SNMPv1-Befehle Antwort und hier auch nur die Systembeschreibung des Agenten. Das Kommandozeilen-Programm „snmpwalk“, das bei fehlender Angabe einer OID im Parameterfeld die „mib-2“ OID einsetzt, liefert lediglich die Managementinformation des Objektes „SNMPv2-MIB::sysDescr.0“. Auf die Managementinformation anderer Objekte hat die „allpublic“-Community überhaupt keinen Zugriff.

6.5.1.2 Konfiguration von Managementinformationen


Listing 6.5.7: Konfiguration von Systeminformationen des Agenten in der Datei „snmpd.conf“

Es gibt auch noch einige andere Anweisungen, welche es erlauben neue Managementinformationen über die Konfigurationsdatei zu definieren. So ist es zum Beispiel auch möglich, Tabellen neue Zeilen mit Managementinformationen anzuhängen oder Netzwerkschnittstellen, die der Net-SNMP Agent bei der Installation nicht erkannt hat, nachträglich zu definieren.
6.5.1.3 Konfiguration der Objekt-Überwachung


Das folgende Beispiel sendet alle ausgehenden SNMP-Meldungen an den Manager mit der IP-Adresse 10.0.0.1 und zwar dreifach, als SNMPv1- und SNMPv2-Trap mit dem Community-Namen „trapCom“, sowie als Inform mit dem Community-Namen „trapatoninie“.

Ohne eine solche Konfigurationsanweisungen verlässt keine SNMP-Meldung den Agenten.

Der Net-SNMP Agent bietet die Möglichkeit die Überwachung mancher Ressourcen des Agenten-Systems, wie zum Beispiel der vorhandenen Speicherkapazität oder der laufenden Anwendungsprozesse, mit wenigen Anweisungen einfach in der Konfigurationsdatei des Agenten zu konfigurieren.

So kann mittels der Anweisung „proc“ die Überwachung der Existenz und Anzahl von Instanzen von laufenden Anwendungsprozessen konfiguriert werden. Dazu wird der Name des Prozesses sowie die Anzahl der maximal und minimal zulässigen Prozessinstanzen angegeben. Der Net-SNMP Agent generiert dann automatisch unterhalb der OID „iso.org.dod.internet.private.enterprises.ucdavis.prTable“ (1.3.6.1.4.1.2021.2) die Objekte bezüglich der konfigurierten Prozesse. Mit der Konfigurationsanweisung „procfix“, die hier allerdings nicht weiter beachtet wird, kann auch angegeben werden was geschehen soll falls die Anzahl der Prozessinstanzen den zulässigen Bereich verlässt.


Werden die beiden Werte der minimal und maximal zulässigen Prozessinstanzen auf 0 gesetzt, so gilt für die Anzahl der mindestens vorhanden Prozessinstanzen 1 und für die der maximal vorhanden unendlich.
Unterhalb der OID des Objektes „prTable“ ist nun die folgende Managementinformation via SNMP zu lesen, wenn von den oben angegebenen Anwendungsprozessen, zwei „ssh“-Dienste, ein „zebra“- und kein „ripd“-Dienst im Agenten-System laufen.

```
Manager # snmptable -v 1 -c private localhost private.enterprises.ucdavis.prTable
SNMP table: UCD-SNMP-MIB::prTable
prIndex  prNames  prMin  prMax  prCount  prErrorFlag  prErrMessage  prErrFix  prErrFixCmd
1        sshd     1      5      2       0           0             0          0
2        ripd     1      1      0       1           Too few ripd running (# = 0) 0
3        zebra    1      1      1       0           0             0          0
```

Befehl 6.20: Abfrage der Prozessüberwachungstable „ucdavis.prTable“


Einen ähnlich einfachen Mechanismus bietet Net-SNMP für das einbinden von Partitionen und der Überwachung der frei verfügbaren Speicherkapazität (OID: ucdavis.dskTable, 1.3.6.1.4.1.2021.9), sowie Dateien und deren maximalen Größe (OID: ucdavis.fileTable, 1.3.6.1.4.1.2021.15).

```
# Disk/File checks
# min free(KB | %)
# path     min free(KB | %)
disk /opt/ 10000
disk /      5%
# filename     max size(KB)
file /tmp/mainlog.log 500
Listing 6.5.10: Konfiguration der Überwachung der Speicherkapazität und Dateigröße
```


Die obigen Konfigurationsanweisungen binden die entsprechenden Ressourcen des Agenten-Systems zur Überwachung in die MIB ein, erzeugen aber nicht automatisch beim überschreiten der Grenzwerte eine SNMP-Meldung. Dies kann über die folgende Konfigurationsanweisung allgemein aktiviert werden. Zuvor aber muss noch über die Anweisung „agentSecName“ ein entsprechender Benutzer generiert werden.

```
# Monitoring
createUser trUser MD5 "trap_password" DES "other_password"
agentSecName trUser
rouser trUser
defaultMonitors yes
Listing 6.5.11: Konfiguration für das Senden von Standard-Ereignismeldungen in der Datei „snmpd.conf“
```

Es können auch noch weitere Überwachungen, die SNMP-Meldungen erzeugen, durch solche Anweisungen aktiviert werden. Das sich zum Beispiel ein Net-SNMP Agent bei seinem Manager meldet, wenn ihn ein SNMP-Befehl mit einem ihm unbekannten Zugriffsnamen erreicht kann über die folgende Konfigurationsanweisung aktiviert werden.

```
authtrapenable 1
Listing 6.5.12: Konfiguration für das Senden einer Meldung bei fehlgeschlagerener Authentifizierung
```


```
# repeat user message-OID message-OID monitored OID value
monitor -r 60 -u trUser -o sysUpTime.0 -o prErrMessage "process" prErrorFlag != 0
Listing 6.5.13: Konfiguration für das Senden von Ereignismeldungen für fehlende Prozessinstanzen
```

6.5.1.4 Konfiguration von Erweiterungen

Net-SNMP bietet die Möglichkeit den Agenten um Sub-Agenten zu erweitern oder externe Bash- oder Perl-Programme direkt einzubinden. Mit der Konfigurationsanweisung „exec“, die in der neueren Net-SNMP Version 5.3.1 durch die Anweisung „extend“ ersetzt wird, können solche externen Programme via SNMP-Befehl aufgerufen und deren Aus- und Rückgabewerte abgefragt werden.

<table>
<thead>
<tr>
<th># executables / scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td># OID</td>
</tr>
<tr>
<td>exec</td>
</tr>
<tr>
<td>exec</td>
</tr>
<tr>
<td>exec</td>
</tr>
<tr>
<td>exec</td>
</tr>
<tr>
<td>exec</td>
</tr>
</tbody>
</table>

Listing 6.5.14: Konfiguration der Einbindung externer Programme über „exec“ in der Datei „snmpd.conf“


Als Beispiel ist hier nun ein kleines Bash-Programm mit dem Namen „testprogie.sh“ beschrieben.

```bash
#!/bin/bash
echo "Hello World"
exit 35
```

Listing 6.6: Beispiel eines Script-Programms „testprogie.sh“ zum Einbinden in den Net-SNMP Agenten

Das Programm gibt lediglich die Textzeile „Hello World“ auf der Konsole aus, wenn es zuvor mit den notwendigen Ausführungsrechten versehen wurde (chmod 755 testprogie.sh).

Agent # /tmp/testprogie.sh
Hello World

Befehl 6.21: Ausgabe des Bash-Scripts „testprogie.sh“

Das Programm „testprogi.sh“ kann über das Abrufen des Objektes „ucdavis.extTable.extEntry.extCommand.1“ (1.3.6.1.4.1.2021.8.1.3.1) via SNMP-Befehl vom Manager aus auf dem Agenten ausgeführt werden. Beim aufrufen der kompletten Tabelle mit allen Managementinformationen des Programms, wie hier gezeigt, wird das Programm „testprogi.sh“ natürlich ebenfalls ausgeführt.

Manager # snmpstable -v 1 -c private 10.0.0.2.1.3.6.1.4.1.2021.2021.xtable
SNMP table: UCD-SNMP-MIB::extTable

```
  extIndex  extNames  extCommand  extResult  extOutput  extErrFix  extErrFixCmd
  1  testprogi  /tmp/testprogi.sh  35  Hello World  0
  2  huuu      /bin/echo   how are you  0  how are you  0

```

Befehl 6.22: Anzeigen der Tabelle der durch „exec“ ausführbaren Programme auf dem Agenten
Net-SNMP in VNUML Simulationen


Im Quellcode-Verzeichnis von Net-SNMP finden sich zwei Beispiel-Programme, das Bash-Programm „passtest“ und das Perl-Programm „pass_persistentest“, die jeweils Managementinformationen über eigene Objekte in einer eigenen „MIB-Bibliothek“ verwalten.

Listing 6.5.15: Konfiguration von extern verwalteten MIB-Bibliotheken über „pass“ und „pass_persist“


Listing 6.7: Beispiel für ein durch „pass“ angesteuertes Bash-Script „vtyshQuery.sh“

Der erste Teil des Programms „vtyshQuery.sh“ definiert die logische Struktur der externen MIB-Bibliothek. Im unteren Teil des Scripts werden dann die Managementinformationen der Objekte definiert.


Diese Objekte können vom Manager aus nur über ihre numerische OID angesteuert werden, so lange keine MIB-Datei mit den SMI-Definitionen der Objekte vorliegt.


### Executables / SMUX Peers

<table>
<thead>
<tr>
<th>#</th>
<th>oid</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>smuxpeer</td>
<td>.1.3.6.1.4.1.3317.1.2.1</td>
<td>quagga_zebra</td>
</tr>
<tr>
<td>smuxpeer</td>
<td>.1.3.6.1.4.1.3317.1.2.2</td>
<td>quagga_bgpd</td>
</tr>
<tr>
<td>smuxpeer</td>
<td>.1.3.6.1.4.1.3317.1.2.3</td>
<td>quagga_ripd</td>
</tr>
<tr>
<td>smuxpeer</td>
<td>.1.3.6.1.4.1.3317.1.2.5</td>
<td>quagga_ospfd</td>
</tr>
<tr>
<td>smuxpeer</td>
<td>.1.3.6.1.4.1.3317.1.2.6</td>
<td>quagga_ospfd6</td>
</tr>
<tr>
<td># smuxsocket tcp:127.0.0.1:199</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Dienst / Abfrage-OID / OID (klartext)

<table>
<thead>
<tr>
<th>Dienst</th>
<th>Abfrage-OID</th>
<th>OID (klartext)</th>
</tr>
</thead>
<tbody>
<tr>
<td>zebra</td>
<td>.1.3.6.1.2.1.4.24</td>
<td>iso.org.dod.internet.mgmt.mib-2.ip.ipForward</td>
</tr>
<tr>
<td>bgpd</td>
<td>.1.3.6.1.2.1.14</td>
<td>iso.org.dod.internet.mgmt.mib-2.bgpd</td>
</tr>
<tr>
<td>ripd</td>
<td>.1.3.6.1.2.1.15</td>
<td>iso.org.dod.internet.mgmt.mib-2.rip2</td>
</tr>
<tr>
<td>ospfd</td>
<td>.1.3.6.1.2.1.23</td>
<td>iso.org.dod.internet.experimental.ospfv3</td>
</tr>
<tr>
<td>ospfd6</td>
<td>.1.3.6.1.3.102</td>
<td>iso.org.dod.internet.experimental.ospfv3</td>
</tr>
</tbody>
</table>

Für die hier aufgeführten Gruppen liefern die Quagga-Autoren MIB-Dateien im Quelltext-Verzeichnis von Quagga mit, so dass vom Net-SNMP Manager aus die Management-Objekte der Router auch über deren Namen angesteuert werden können. Die MIB-Dateien müssen in das entsprechende Standard-MIB Verzeichnis kopiert und über die Umgebungsvariable MIBS (export MIBS="ALL") den Net-SNMP Programmen bekannt gemacht werden. Natürlich muss SMUX auch in den Konfigurationsdateien der Routing-Dienste eingestellt werden, was in Kapitel 5.7.2 behandelt wurde.

Darüber hinaus gibt es noch weitere Konfigurationsanweisungen mit denen die Kommunikation mit dem AgentX Sub-Agenten detaillierter eingestellt werden kann.

```
# executables / AgentX
#agentXSocket tcp:127.0.0.1:705
Listing 6.5.17: Konfiguration des AgentX-Sub-Agenten in der Datei „snmpd.conf“
```

Auch der Net-SNMP Agent selbst kann als AgentX Sub-Agent gestartet werden, so dass er sich über das Erweiterungsprotokoll an einen Master SNMP-Agenten ankoppelt.

```
Agent # snmpd -X-x localhost:705 -c /etc/snmp/subxd.conf -Lf /var/log/subxd.log
Befehl 6.24: Starten des Net-SNMP Agenten als AgentX-Sub-Agent
```


### 6.5.2 Die Anwendungsprogramme des Net-SNMP Agenten


**snmptrap, snmpinform:**

```
Agent # snmptrap -v1 -c publicTrap 10.0.0.1 "" 0 0 ""
Agent # snmpinform -v2c -c publicTrap 10.0.0.1 "" UCD-SNMP-MIB::ucdStart
Befehl 6.25: Versenden von SNMP-Meldungen
```

Wie bei den Kommandozeilen-Programmen des Managers ist auch hier der Vorteil, dass die Programme in komplexere Scripte, wie Bash- oder Perl-Scripte, eingebunden werden können.

**net-snmp-config:**

```
Agent # net-snmp-config --compile-subagent myagentx_name agentx.c
Befehl 6.26: Kompilieren eines AgentX-Sub-Agenten für Net-SNMP
```

6.5.3 Erstellen und Einbinden eigener MIB-Module

Im vorherigen Kapitel 6.5.1 ging es bereits um die Konfiguration des Net-SNMP Agenten. Dabei wurde gezeigt wie über Anweisungen in der Konfigurationsdatei einige Managementinformationen für die Verwaltung via SNMP in den Agenten eingebunden werden können. Diese Möglichkeit der Einbindung von Managementinformationen ist zwar schnell durchgeführt und benötigt keine größeren Hintergrundkenntnisse, ist aber auch relativ stark auf die Vorgaben von Net-SNMP beschränkt.


Um Ressourcen des Agenten-Systems zu implementieren müssen die folgenden Dateien erstellt werden:

- eine MIB-Datei, die die SMI-Definition des Objektes der Ressource (Managementinformation) enthält
- eine C-Header-Datei (dateiname.h)
- eine C-Anwendungs-Datei (dateiname.c)


Das Programm „mib2c“ generiert auf Basis der SMI-Objekt-Definitionen einer MIB-Datei eine C-Header-Datei und eine C-Anwendungs-Datei, deren Inhalt diese C-Quellcode Vorlage ist. Die erstellte C-Header-Datei muss nicht mehr verändert werden, wird aber aber später für den Kompiliervorgang benötigt. Die C-Anwendungs-Datei beinhaltet nach ihrer Erstellung bereits sämtliche notwendigen Funktionen der Net-SNMP C-API, um ein neues Management-Objekt einzubinden. Ihr fehlt allerdings der „funktionalen“ C-Code, also der Teil des Quellcodes der den eigenen Zugriff. Über diese C-Quellcode muss in die C-Anwendungs-Datei, an den entsprechenden Stellen, eingefügt werden.

Somit sind zwar Grundkenntnisse der Programmiersprache C notwendig, allerdings nur ein minimales Wissen über die Funktionen der Net-SNMP C-API.


Über das zweite Management-Objekt, mit Namen „ipv4Forwarding“ (ID: 2), soll die IPv4-Weiterleitung eines Agenten-Systems und damit dessen Funktion als Durchganges-Server, via SNMP von einem Manager aus, an- und abgeschaltet werden können. Ist die Funktion abgeschaltet so leitet das Agenten-System keine eingehenden Netzwerk-Nachrichten weiter, die an externe IPv4-Adressen gerichtet sind. Dieses Objekt ist eine Auswahlmöglichkeit über dem Datentyp „INTEGER“, wobei der Wert 0 mit der Auswahl „nonIpForwarding“, also IP-Weiterleitung aus, und der Wert 1 mit der Auswahlmöglichkeit „IpForwarding“, also IP-Weiterleitung...

Der Teilbaum der „Management Information Base“ sieht nach Eingliederung der Objekte wie folgt aus.


---

```
STUDIENARBEIT-MIB DEFINITIONS ::= BEGIN

IMPORTS
   enterprises, MODULE-IDENTITY, OBJECT-TYPE, INTEGER
   DisplayString FROM SNMPv2-SMI
   DisplayString FROM SNMPv2-TC;

--
-- A brief description and update information about this mib.
--
studienarbeit MODULE-IDENTITY
   LAST-UPDATED "0411060000Z" -- 11 April 2006, midnight
   ORGANIZATION " - "
   CONTACT-INFO "
   Author: Frank Bohdanowicz, Studienarbeit: SNMP in VNUML Simulationen,
           AG Rechnernetze, University of Koblenz, 56070 Koblenz, Germany,
           email: bohdan@uni-koblenz.de, phone: - "
   DESCRIPTION "Example MIB for remote control via SNMP"
::= { enterprises 9876 }

server-mgmt OBJECT IDENTIFIER ::= { studienarbeit 1 }
reserved OBJECT IDENTIFIER ::= { studienarbeit 2 }

-- a command of max 255 bytes length
--
superUserCommand OBJECT-TYPE
   SYNTAX DisplayString (SIZE (0..255))
   MAX-ACCESS read-write
   STATUS current
   DESCRIPTION "A variable command, executed as root user"
::= { server-mgmt 1 }

-- an integer value 0 or 1 switches IP-Forwarding on/off
--
ipv4forwarding OBJECT-TYPE
   SYNTAX INTEGER {
      nonIpForwarding(0), -- forwarding off
      ipForwarding(1) -- forwarding on
   }
   MAX-ACCESS read-write
   STATUS current
   DESCRIPTION "Switches ipv4 Forwarding on/off"
::= { server-mgmt 2 }
END
```

---

Abb. 6.2: MIB-Modul „studienarbeit“ im ISO-Registrierungsbau

Listing 6.9: Inhalt der MIB-Datei „STUDIENARBEIT-MIB.txt“

Folglich sind also folgende Schritte zu erledigen.

Nun sollten die neuen Management-Objekte über den Befehl „snmptranslate“ einzusehen sein.

```
Agent # cp STUDIENARBEIT-MIB.txt /usr/local/share/snmp/mibs/
Agent # cp STUDIENARBEIT-MIB.txt /usr/share/snmp/mibs/
Agent # export MIBS="ALL"
```

Befehl 6.27: Eingliederung der „STUDIENARBEIT-MIB“ in die Net-SNMP MIB-Umgebung

```
Agent # snmptranslate -Tp -IR studienarbeit
 +--studienarbeit(9876)
   |   +--server-mgmt(1)
   |   |   +--RW- String superUserCommand(1)
   |   |   |   Textual Convention: DisplayString
   |   |   |   Size: 0..255
   |   |   +--RW- EnumVal ipv4forwarding(2)
   |   |   |   Values: nonIpForwarding(0), ipForwarding(1)
   |   +--reserved(2)
```

Befehl 6.28: Anzeigen der neuen Objekte mit „snmptranslate“

Ist dies der Fall, so kann nun die C-Anwendungs-Datei und die C-Header-Datei erstellt werden. Da innerhalb der „STUDIENARBEIT-MIB“ lediglich Skalare-Objekte definiert sind, kann als Stil-Vorlage die Datei „mib2c.scalar.conf“ zum Erstellen der C-Quellcode-Vorlage verwendet werden.

Die beiden benötigten C-Dateien können dann wie folgt, in einem beliebigen Verzeichnis, erstellt werden.

```
Agent # mib2c -c/usr/local/share/snmp/mib2c.scalar.conf studienarbeit
writing to studienarbeit.h
writing to studienarbeit.c
running indent on studienarbeit.h
running indent on studienarbeit.c
```

Befehl 6.29: Erstellen der C-Dateien auf Basis der STUDIENARBEIT-MIB

Wie an der Ausgabe zu erkennen ist werden hier die beiden C-Dateien erzeugt und innerhalb des Verzeichnisses abgelegt. Der Befehl „mib2c studienarbeit“, ohne Angabe einer Stil-Vorlage, erstellt die C-Dateien nicht sofort wie hier, sondern öffnet einen Frage/Antwort-Dialog, über den die passende Stil-Vorlage gewählt werden kann.

Die erstellte C-Header-Datei „studienarbeit.h“ kann so wie sie ist belassen werden. Die C-Anwendungs-Datei „studienarbeit.c“ beinhaltet nun bereits alle für die SNMP Kommunikation notwendigen C-Funktionen der Net-SNMP C-API und muss nun noch um die C-Funktionen, die auf die Objekt-Ressource des Agenten-Systems zugreifen, erweitert werden.
Im folgenden wird nun Stück für Stück die fertige C-Anwendungs-Datei erläutert. Die Code-Stücke die zusätzlich eingefügt wurden sind dabei in Fett-Schrift markiert, alle anderen Code-Stücke sind zuvor automatisch vom „mib2c-Parser“ eingefügt worden.


Als nächstes folgt mit der Initialisierung schon der Kern der C-Anwendungsdatei. In dieser „init“-Funktion werden sämtliche Management-Objekte beim Net-SNMP Agenten registriert. Über die Funktion „netsnmp_register_scalar“ wird ein Skalar-Objekt eingebunden. Angegeben werden muss dabei, der Name des Objektes, die C-Funktion die bei Abruf aufgerufen werden soll, die OID des Objektes, die Bytelänge der OID und die Zugriffsrechte, die das Lesen oder Schreiben des Objektes festlegen.

Hier ist nichts mit Fett-Schrift markiert da das Programm „mib2c“ die komplette Funktion selbst erstellt. Allerdings ist darauf zu achten, dass das Programm die richtige OID einsetzt. Da die Objekte aus der globalen Net-SNMP „Management Information Base“ gelesen werden, kann es bei gleichen textuellen Objekt-Bezeichnungen zu Verwechslungen kommen. Anschließend folgt auch schon die Definition der Funktionen, die bei Abruf des Objektes aufgerufen werden sollen.

Hier folgt der Kopf der ersten Funktion, die für das Objekt „superUserCommand“ den Zugriff implementiert.


Als letzten Fall der Switch-Case Anweisung, gibt es noch „DEFAULT“. Dieser Fall gilt aber als tiefergehender Fehlerzustand des Net-SNMP Agenten und sollte nach Möglichkeit nie erreicht werden.


```
switch (reqinfo->mode) {
    case MODE_GET:
        snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
            (char *)&superUserCommand, sizeof(superUserCommand));
        break;

    case MODE_SET_RESERVE1:
        break;
    case MODE_SET_RESERVE2:
        break;
    case MODE_SET_FREE:
        break;
    case MODE_SET_ACTION:
        sprintf(superUserCommand, "%s", (requests->requestvb->val.string));
        system(superUserCommand);
        break;

    case MODE_SET_COMMIT:
        break;
    case MODE_SET_UNDO:
        break;

    default:
        snmp_log(LOG_ERR,
            "unknown mode (%d) in handle_superUserCommand\n",
            reqinfo->mode);
        return SNMP_ERR_GENERR;
    }
    return SNMP_ERR_NOERROR;
}
```

Listing 6.10.4: Inhalt der Datei „studienarbeit.c“, Teil 4

Net-SNMP in VNUML Simulationen


Direkt im Anschluss folgt dann auch schon die Funktion, die das nächste Objekt „ipv4forwarding“ verwaltet. Das Grundgerüst dieser Funktion ist gleich mit dem der Funktion zuvor.

In der Funktion „handle_ipv4forwarding“ wird im Fall „MODE_GET“ der Switch-Case Anweisung der Inhalt der Datei „ip_forward“, die sich im Verzeichnis „/proc/sys/net/ipv4/“ befindet, ausgelesen, in die Integer Variable ipForwarding geschrieben und dann zum senden an die Net-SNMP-Agenten Funktion „snmp_set_var_typed_value“ übergeben. Die Datei „ip_forward“ enthält normalerweise den Wert 0 oder 1, was jeweils angibt, ob die Weiterleitung von Nachrichten an IP-Adressen anderer Rechner aus- oder eingeschaltet ist. Es wird also der aktuelle Zustand direkt aus dem Agenten-System gelesen.

Im Fall MODE_SET_ACTION wird der empfangene Objekt-Wert in die Variable „ipForwarding“ geschrieben, anschließend zusammen mit dem Linux Befehl „sysctl“ über die C-Funktion „sprintf“ in der Char-Array Variable „sysforward“ verknüpft und dann über die C-Funktion „system“ auf dem Agenten-System ausgeführt.

```c
int handle_ipv4forwarding(netsnmp_mib_handler *handler,
    netsnmp_handler_registration *reginfo,
    netsnmp_agent_request_info *reqinfo,
    netsnmp_request_info *requests)
{
    switch (reqinfo->mode) {
    case MODE_GET:
        stream = fopen("/proc/sys/net/ipv4/ip_forward","r");
        if(stream != NULL){ fscanf(stream,"%d",&ipForwarding); }
        fclose(stream);
        snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
            (u_char *)&ipForwarding,
            sizeof(ipForwarding));
        break;
    case MODE_SET_RESERVE1:
        break;
    case MODE_SET_RESERVE2:
        break;
    case MODE_SET_FREE:
        break;
    case MODE_SET_ACTION:
        ipForwarding = *(requests->requestvb->val.integer);
        char sysforward[]="sysctl net.ipv4.ip_forward=
        sprintf(sysforward,"%s%d",sysforward,ipForwarding);
        system(sysforward);
        break;
    case MODE_SET_COMMIT:
        break;
    case MODE_SET_UNDO:
        break;
    default:
        snmp_log(LOG_ERR, "unknown mode (%d) in handle_ipv4forwarding\n", reqinfo->mode);
        return SNMP_ERR_GENERR;
    }
    return SNMP_ERR_NOERROR;
}
```

Listing 6.10.5: Inhalt der Datei „studienarbeit.c“, Teil 5

Damit wäre die C-Anwendungs-Datei „studienarbeit.c“ fertig und muss nun noch kompiliert werden.
Zwei Möglichkeiten die Dateien zu kompilieren und die Management-Objekte in einen Net-SNMP Agenten einzubinden werden hier nun aufgezeigt.

Die erste Möglichkeit ist die direkte Implementierung in den Net-SNMP Agenten, was während des Kompilierers vor der Installation geschehen muss. Die C-Header Datei „studienarbeit.h“ und die C-Anwendung-Datei „studienarbeit.c“ müssen in das Verzeichnis „agent/mibgroup“, das sich innerhalb des Net-SNMP Quelltext Verzeichnisses befindet, kopiert werden.


Die zweite Möglichkeit ist das Kompilieren zu einem eigenständigen AgentX Sub-Agentenprogramm, mittels dem bereits vorgestellten Tool „net-snmp-config“. Der Vorteil hierbei ist, dass die Management-Objekte nur dann von einem Manager aus ansteuerbar sind, wenn dieses Programm läuft. Außerdem muss nicht das komplette Net-SNMP Softwarepaket neu kompiliert und installiert werden. Laut Net-SNMP-Entwicklergruppe ist diese Methode allerdings noch nicht ganz ausgereift und es kann zu Problemen kommen. Während der praktischen Ausarbeitung der Studienarbeit traten jedoch keinerlei Probleme auf, was aber daran liegen kann, dass nur einfache SMI-Konstrukte verwendet wurden.

Das Programm muss in dem Verzeichnis ausgeführt werden, indem sich die C-Anwendungs- und die C-Header-Datei befinden. Der folgende Befehl kompiliert aus den zuvor erstellten Dateien das Programm mit Namen „studienProgie“, das anschließend im gleichen Verzeichnis zu finden ist.


Solange das Programm läuft können die beiden Management-Objekte verwendet werden.


Zum Beispiel können so nun neue Routen-Konfigurationen, über den Befehl „route“, vom Manager aus auf dem Agenten-System eingestellt werden. Der folgende Befehl fügt einen Eintrag in die Kernel Routing Tabelle des Agenten-Systems hinzu, der den Agenten das Netz 192.168.0.0/30 mit einer Metric von fünf über die Netzwerkschnittstelle eth1 erreichen lässt.
Der Agent besitzt nach dem Senden des Befehls diese Route was mit dem Befehl „route -n“ über die Kernel-Tabelle festgestellt werden kann.

<table>
<thead>
<tr>
<th>Agent # route -n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel IP Routentabelle</td>
</tr>
<tr>
<td>Ziel</td>
</tr>
<tr>
<td>192.168.0.4</td>
</tr>
<tr>
<td>10.0.0.0</td>
</tr>
<tr>
<td>192.168.0.0</td>
</tr>
<tr>
<td>127.0.0.0</td>
</tr>
</tbody>
</table>

Befehl 6.34: Überprüfen der Netzwerk-Routen mit dem Befehl „route“


<table>
<thead>
<tr>
<th>Manager # snmpset -v1 -c private 10.0.0.2 studienarbeit.1.1.0 &quot;ip route add 192.168.0.0/30 via 10.0.0.1“</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDIENARBEIT-MIB::superUserCommand.0 = STRING: ip route add 192.168.0.0/30 via 10.0.0.1</td>
</tr>
</tbody>
</table>

Befehl 6.35: Übergeben einer neuen IP-Route via SNMP über das Objekt „superUserCommand“


<table>
<thead>
<tr>
<th>Manager # snmpset -v1 -c public -Cw 110 10.0.0.2 ip.ipRouteTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP table: RFC1213-MIB::ipRouteTable</td>
</tr>
<tr>
<td>ipRouteDest</td>
</tr>
<tr>
<td>10.0.0.0</td>
</tr>
<tr>
<td>127.0.0.0</td>
</tr>
<tr>
<td>192.168.0.0</td>
</tr>
<tr>
<td>192.168.0.4</td>
</tr>
</tbody>
</table>

SNMP table RFC1213-MIB::ipRouteTable, part 2

<table>
<thead>
<tr>
<th>ipRouteType</th>
<th>ipRouteProto</th>
<th>ipRouteAge</th>
<th>ipRouteMask</th>
<th>ipRouteMetric5</th>
<th>ipRouteInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct</td>
<td>local</td>
<td>?</td>
<td>255.255.255.0</td>
<td>?</td>
<td>SNMPv2-SMI::zeroDotZero</td>
</tr>
<tr>
<td>direct</td>
<td>local</td>
<td>?</td>
<td>255.0.0.0</td>
<td>?</td>
<td>SNMPv2-SMI::zeroDotZero</td>
</tr>
<tr>
<td>indirect</td>
<td>local</td>
<td>?</td>
<td>255.255.255.252</td>
<td>?</td>
<td>SNMPv2-SMI::zeroDotZero</td>
</tr>
<tr>
<td>indirect</td>
<td>local</td>
<td>?</td>
<td>255.255.255.252</td>
<td>?</td>
<td>SNMPv2-SMI::zeroDotZero</td>
</tr>
</tbody>
</table>

Befehl 6.36: Auslesen der Routing-Tabelle des Agenten via SNMP


<table>
<thead>
<tr>
<th>Agent # sysctl net.ipv4.ip_default_ttl</th>
</tr>
</thead>
<tbody>
<tr>
<td>net.ipv4.ip_default_ttl = 64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Agent # sysctl net.ipv4.ip_default_ttl=16</th>
</tr>
</thead>
<tbody>
<tr>
<td>net.ipv4.ip_default_ttl = 16</td>
</tr>
</tbody>
</table>

Befehl 6.37: Konfiguration der Netzwerkschnittstellen mit dem Befehl „sysctl“

Als Eingabe folgt dem Befehl die Datei mit dem zu konfigurierenden Wert, wobei die einzelnen Unterzeichenisse bis zur Datei durch Punkte getrennt werden müssen.
Statt den Befehl direkt in der Konsole des Agenten einzugeben kann er nun auch via SNMP über das Objekt „superUserCommand“ vom Manager aus an den Agenten gesendet und dort ausgeführt werden.

Manager # snmpset -v1 -c private 10.0.0.2 studienarbeit.1.1.0 s 'sysctl net.ipv4.ip_default_ttl=16'
STUDIENARBEIT-MIB::superUserCommand.0 = STRING: sysctl net.ipv4.ip_default_ttl=16

Befehl 6.38: Übermittlung des „sysctl“ Befehls an den Agenten via SNMP


Manager # snmpset -v1 -c private 10.0.0.2 studienarbeit.1.2.0 i ipForwarding
STUDIENARBEIT-MIB::ipv4forwarding.0 = INTEGER: ipForwarding(1)

Befehl 6.39: Abfrage IPv4-Weiterleitung über das „ipv4forwarding“ Objekt

Das dargestellte Abfrage-Beispiel ist so nur ausführbar, wenn der Manager auch Zugriff auf die MIB-Datei „STUDIENARBEIT-MIB.txt“ hat. Über die dort enthaltene SMI-Definition des Objektes wird auch der Begriff „ipForwarding“ in den entsprechende Integerwert, in diesem Fall 1, abgebildet. Kann der Manager nicht auf die MIB-Datei zugreifen, so ist ein Zugriff nur über die Eingabe der numerischen Werte möglich.

Manager # snmpset -v1 -c private 10.0.0.2 .1.3.6.1.4.1.9876.1.2.0 i 0
iso.3.6.1.4.1.9876.1.2.0 = INTEGER: 0

Befehl 6.40: Setzen der IPv4-Weiterleitung via SNMP über die numerischen Werte

Hier wird auch das Problem von SNMP offenbar, dass für eine komfortable und fehlerfreie Anwendung die MIB-Dateien auf dem Manager nicht nur verfügbar, sondern auch in der gleichen Version wie deren Implementierung auf dem Agenten vorhanden sein müssen, wenn der Manager die numerische OID des Objektes nicht kennen soll oder kann.

6.6 Anwendungsbeispiele


Das Bash-Programm wird dann beispielsweise wie folgt gestartet.

```
Manager # /ifTraffic.sh 10.0.0.2 public eth1 ifTrafficAgent_eth1.csv
```


```
Name: Agent , Interface: eth1 
ifInOctets , ifOutOctets , ifInUcastPkts , ifOutUcastPkts 
29458 ,  29458 ,  249 ,  249 , 
29848 ,  29848 ,  251 ,  251 , 
30238 ,  30238 ,  253 ,  253 , 
30628 ,  30628 ,  255 ,  255 , 
31018 ,  31018 ,  257 ,  257 , 
31408 ,  31408 ,  259 ,  259 , 
31798 ,  31798 ,  261 ,  261 ,
```

Eine solche CSV Datei kann zum Beispiel mit Tabellenkalkulationsprogrammen, auch grafisch, weiterverarbeitet und ausgewertet werden.


```
#!/bin/bash
NULL=0
TRUE=1
while [ "$TRUE" ]
do
  set -- `ps -A | grep -c zebra`
  RUN=$1
  if [ "$RUN" == "$NULL" ]
    then
      set -- `snmptrap -v2c -c trapatonie 10.0.0.1
             ZEBRA-NOTIFY-MIB::noZebra`
    fi
  sleep 30
done
exit
```

```
Listing 6.13: Inhalt der Script-Datei „zebraTest.sh“
```


```
ZEBRA-NOTIFY-MIB DEFINITIONS ::= BEGIN
IMPORTS
  NOTIFICATION-TYPE FROM SNMPv2-SMI
  studienarbeit reserved FROM STUDIENARBEIT-MIB;

notify OBJECT IDENTIFIER ::= { reserved 1 }

noZebra NOTIFICATION-TYPE
  OBJECT { INTEGER }
  STATUS current
  DESCRIPTION "no running Zebra"
  ::= { notify 123 }
END
```

```
Listing 6.14: Definition des Notification-Objects „noZebra“
```

```
Befehl 6.42: „snmptranslate“ über „studienarbeit“
```

Auf der rechten Seite ist eine leicht verkürzte Ausgabe des Befehls „snmptranslate -Tp -IR studienarbeit“ zusehen, nachdem die MIB-Datei eingebunden wurde.


\begin{verbatim}
2006-04-28 00:14:34 localhost [UDP: [10.0.0.2]:32808]:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (5840489) 16:13:24.89
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9876.2.1.123
\end{verbatim}

\textbf{Listing 6.15: Inhalt der Log-Datei des Trap-Servers nach Erhalt der „noZebra”-Meldung}


\begin{verbatim}
trapcommunity trapatonie
traphandle ZEBRA-NOTIFY-MIB::noZebra /tmp/zebra.sh down
\end{verbatim}

\textbf{Listing 6.16: Konfiguration der „snmptrapd.conf“ für den Empfang der Ereignismeldung „noZebra“}


\begin{verbatim}
#!/bin/bash
read host
read ip
vars=
while read oid val
do
if [ "$vars" == "" ]
then
vars="$oid = $val"
else
vars="$vars, $oid = $val"
fi
done
echo "trap: $1 $host $ip $vars" >> zebraOutput.txt
\end{verbatim}

\textbf{Listing 6.17: Inhalt des Bash-Programms „zebra.sh“}

In die Datei „zebraOutput.txt“ werden durch das Bash-Programm Einträge der folgenden Art gemacht.

\begin{verbatim}
trap: down localhost UDP: [10.0.0.2]:32808 DISMAN-EVENT-MIB::sysUpTimeInstance = 0:16:43:50.31, SNMPv2-MIB::snmpTrapOID.0 = ZEBRA-NOTIFY-MIB::noZebra
\end{verbatim}

\textbf{Listing 6.18: Inhalt der Datei „zebraOutput.txt“}


Bei der Verwendung von Net-SNMP in VNUML Netzwerk-Simulationen konnte festgestellt werden, dass die SNMP-Nachrichten insgesamt durchaus einen sehr hohen Datenverkehr verursachen, auch wenn die einzelne SNMP-Nachricht recht klein ist. Je mehr Managementinformationen mit einer einzelnen Nachricht übertragen werden können, um so effizienter aber meist auch unflexibler ist der Einsatz von SNMP. Deswegen ist beim praktischen Einsatz von SNMP gerade im Sinne des Leistungsmanagements nicht nur die Managementinformation, die abgefragt werden soll und das Abfrage-Intervall ein wichtiger Faktor, sondern auch die technische Umsetzung über die Anzahl der verwendeten SNMP-Nachrichten.


<table>
<thead>
<tr>
<th>Versandart der Objekte</th>
<th>SNMP Version</th>
<th>Auswertungszeit (sekunden)</th>
<th>Anzahl gesendeter Nachrichten</th>
<th>gesendete Datenmenge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Jedes Objekt über eine eigene Nachricht</strong></td>
<td>SNMPv1/-v2c</td>
<td>real: ~ 0,960 s</td>
<td>6</td>
<td>705 Byte</td>
</tr>
<tr>
<td></td>
<td>SNMPv3 (noAuthNoPriv)</td>
<td>real: ~ 0,970 s</td>
<td>12</td>
<td>1996 Byte</td>
</tr>
<tr>
<td></td>
<td>SNMPv3 (authNoPriv)</td>
<td>real: ~ 1,050 s</td>
<td>12</td>
<td>2061 Byte</td>
</tr>
<tr>
<td></td>
<td>SNMPv3 (authPriv)</td>
<td>real: ~ 1,160 s</td>
<td>12</td>
<td>2164 Byte</td>
</tr>
<tr>
<td><strong>Alle Objekte über eine einzige Nachricht</strong></td>
<td>SNMPv1/-v2c</td>
<td>real: ~ 0,340 s</td>
<td>2</td>
<td>359 Byte</td>
</tr>
<tr>
<td></td>
<td>SNMPv3 (noAuthNoPriv)</td>
<td>real: ~ 0,350 s</td>
<td>4</td>
<td>791 Byte</td>
</tr>
<tr>
<td></td>
<td>SNMPv3 (authNoPriv)</td>
<td>real: ~ 0,380 s</td>
<td>4</td>
<td>815 Byte</td>
</tr>
<tr>
<td></td>
<td>SNMPv3 (authPriv)</td>
<td>real: ~ 0,400 s</td>
<td>4</td>
<td>845 Byte</td>
</tr>
</tbody>
</table>

Tabelle 6.1: Vergleich des Ressourcenverbrauchs der einzelnen SNMP-Versionen

6.7 Zusammenfassung


Der Vorteil von Net-SNMP im allgemeinen ist die sehr gute Unterstützung von Linux-Systemen. Es können zahlreiche Managementinformationen über das Linux System und seinen Zustand direkt nach der Installation via SNMP abgefragt werden.


Net-SNMP kann in VNUML-Simulationen jedoch auch sehr gut dazu verwendet werden SNMP-Manager-Anfragen auszuarbeiten und für weitestgehend beliebige Netzwerk-Topologien zu Optimieren, um zum Beispiel den SNMP-Datenverkehr so effizient und so klein wie möglich zu gestalten.

Anhang: Anleitung

7 Erstellen eines VNUML-Rechners auf Basis von Gentoo-Linux

Im Laufe der praktischen Ausarbeitung dieser Studienarbeit wurde ein UML-Root-Dateisystem auf Basis der Gentoo-Linux Distribution sowie ein UML-Kernel für den Einsatz in VNUML Netzwerk-Simulationen erstellt. In diesem Kapitel wird die Vorgehensweise dabei aufgezeigt.

7.1 Einleitung


Die Installation von Net-SNMP und Quagga, insbesondere die manuelle Installation über den Quellcode, war jedoch nur sehr umständlich und nicht mit den gewünschten Konfigurationen zu realisieren.


- http://www.gentoo.org - Gentoo Projekt Homepage
- http://www.gentoo.de - Deutsche Gentoo Projekt Homepage
- http://gentoo-wiki.com - Zahlreiche Anleitungen für das Einrichten verschiedener Programme
- http://gentoo-portage.com - Erläuterungen zu Softwarepaketen und deren Konfigurationsmöglichkeiten

7.2 Erstellen eines VNUML-Root-Dateisystems auf Basis von Gentoo-Linux

7 Erstellen eines VNUML-Rechners auf Basis von Gentoo-Linux

„Stage 3“ beinhaltet ein komplettes lauffähiges, minimales Gentoo-Linux Basis-System. „Stage 1“ verlangt vom „Bootstrapping“ angefangen das Kompilieren aller notwendigen Systemanwendungen für die Funktion, was letztlich ein perfekt an die eigene Hardware und eigenen Bedürfnisse angepasstes Betriebssystem erlaubt. Mit „Stage 1“ kann das komplette Linux-System aus den Quelltexten gebaut werden, mit individuell eingestellten Optimierungen. „Stage 2“ liegt mit dem Aufwand für das Kompilieren von Systemanwendungen zwischen Stage 1 und 3.


7.2.1 Vorbereiten des Gentoo-Systems

Für die Erstellung eines UML-Root-Dateisystems auf Basis von Gentoo-Linux für den Einsatz in VNUML müssen auf dem Host-System folgende Voraussetzungen erfüllt sein:

- ausreichende Zugriffsrechte auf dem Host-System, am besten root-Rechte (SuperUser)
- bestehende Internetverbindung
- ca. 3 Gigabyte freien Speicherplatz


Als Gentoo-Linux Basis-System genügt hier ein „Stage3“-System da es nicht erst zeitaufwendig bearbeitet werden muss um als UML-Root-Dateisystem für VNUML oder einen einzelnen UML-Rechner eingesetzt zu werden. Ein entsprechender „Stage3“-Tarball ist auf den Gentoo-Servern im Verzeichnis „gentoo/releases/Architektur/current/stages“ zu finden. Der Bezeichner 'Architektur' steht für die Architektur des Rechners, die wegen der Virtualisierung aber eher unerheblich sein dürfte. Um Probleme zu vermeiden sollte aber trotzdem die Architektur des Host-Rechners verwendet werden.


Mit dem Download-Programm „wget“ werden nun die benötigten Tarballs von einem Spiegel-Server aus dem Internet auf das lokale Host-System heruntergeladen und im Verzeichnis „download“ gespeichert.

<table>
<thead>
<tr>
<th>host</th>
<th>mkdir download</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>cd download</td>
</tr>
<tr>
<td>host download</td>
<td>wget ftp://ftp.uni-koblenz.de/pub/mirrors/gentoo/releases/x86/current/stages/stage3-x86-2006.0.tar.bz2</td>
</tr>
<tr>
<td>host download</td>
<td>wget ftp://ftp.uni-koblenz.de/pub/mirrors/gentoo/snapshots/portage-latest.tar.bz2</td>
</tr>
<tr>
<td>host download</td>
<td>cd ..</td>
</tr>
</tbody>
</table>

Befehl 7.1: Download des „Stage3“- und des „Portage“-Tarballs

Anschließend folgt das Entpacken der beiden Tarballs.
Der Tarball, der das Gentoo-Linux Basis-System enthält, kann einfach in ein Verzeichnis entpackt werden, hier in das Verzeichnis „gentoo_backup“.

```
host # mkdir gentoo_backup
host # tar xfjvp download/stage3-x86-2006.0.tar.bz2 -C gentoo_backup/
```

Befehl 7.2: Entpacken des „Stage3“-Tarballs

In dem Verzeichnis „gentoo_backup“ befindet sich nun schon ein verwendbares UML-Root-Dateisystem, das für den Einsatz in VNUMUL aber noch entsprechend konfiguriert werden muss.


```
host # dd if=/dev/zero of=portage.img seek=2000 count=1 bs=1M
host # mke2fs -F portage.img
host # mkdir -p gentoo_backup/usr/portage
host # mount -o loop portage.img gentoo_backup/usr/portage
host # tar xfjvp download/portage-latest.tar.bz2 -C gentoo_backup/usr/
```

Befehl 7.3: Entpacken des „Portage“-Tarballs

Um aus dem UML-Root-Dateisystem heraus auch auf Internet-Adressen zugreifen zu können, muss noch die Datei „resolv.conf“ des lokalen „Resolvers“ aus dem Host-System in das Gentoo-System kopiert werden.

```
host # cp /etc/resolv.conf gentoo_backup/etc/
```

Befehl 7.4: Kopieren der Datei „resolv.conf“ für den Zugriff auf Internet-Adressen


```
host # mount -t proc proc gentoo_backup/proc
host # chroot gentoo_backup /bin/bash
/# env-update && source /etc/profiles
>> Regenerating /etc/ld.so.cache... 
```

Befehl 7.5: Wechsel in die „chroot“-Umgebung in das Gentoo-Linux-System

Damit kann nun mit der Konfiguration des Gentoo-Systems begonnen werden.

### 7.2.2 Konfiguration des Gentoo-Systems für VNUML


```
/# passwd
New UNIX password: xxxxx
BAD PASSWORD: it is too short
Retype new UNIX password: xxxxx
password: password updated successfully
```

Befehl 7.6: Setzen eines Passwortes für den UML-Rechner Benutzer „root“

Die Einstellungen in der Datei „/etc/fstab“ gehören zu den wichtigsten Konfigurationen des UML-Rechners. Ist die Datei falsch konfiguriert, so funktioniert der UML-Rechner nicht.

VNUML in der Version 1.5 verlangt nun noch die Erstellung eines „Mountpunktes“ für die zweite Partition, hier das Verzeichnis „/mnt/vnuml/“, sowie für die Funktion des VNUML-Managements via SSH notwendig, einen Link auf dieses Verzeichnis vom SSH-Verzeichnis des UML-Benutzers „root“ aus.

Für die automatische, individuelle Einrichtung der UML-Rechner der Netzwerk-Simulationen geht VNUML so vor, dass es innerhalb der zweiten Partition (/dev/ubd/1) eine Script-Datei erstellt, die beim Start des UML-Rechners ausgeführt wird und diesen, entsprechend den Vorgaben der Szenario-Beschreibung, konfiguriert. Eine genauere Erläuterung der Vorgehensweise ist in Kapitel 5.5 zu finden.

Damit die Script-Datei beim Start des UML-Rechners ausgeführt wird, muss sie aus einem Runlevel heraus referenziert sein. Hierfür muss ein Link aus dem Default-Runlevel auf die Datei „/mnt/vnuml/umlboot“ gesetzt werden, was in Gentoo-Linux ein Init-Script voraussetzt. Dieses Init-Script, hier genannt „vnumlboot“, muss im Verzeichnis „/etc/init.d/“ abgelegt sein und den folgenden Inhalt besitzen.

```
#!/sbin/runscript

depend() {
  need localmount
  use net
}

start() {
  ebegin "umlboot starts network interfaces config"
  if [ ! -x /mnt/vnuml/umlboot ]
    then
      error "Unable to locate /mnt/vnuml/umlboot script !"
      return 1
    fi
  /mnt/vnuml/umlboot
  retval=$?
  eend $retval "Failed to run /mnt/vnuml/umlboot script !"
}
```

Dieses Init-Script muss ausführbar sein und über den Befehl „re-update“ in das Default-Runlevel gesetzt werden.

```
#/chmod 755 /etc/init.d/vnumlboot
#/ re-update add vnumlboot default
* vnumlboot added to runlevel default
*re-update complete
```

Listing 7.1: Konfiguration der Datei „/etc/fstab“

---

**Befehl 7.7: Erstellen des „Mountpunktes“ für die zweite Partition und SSH-Verlinkung**

```
# mkdir /mnt/vnuml
# ln -sF /mnt/vnuml /root/.ssh
```

---

**Befehl 7.8: Setzen des Init-Scripts „vnumlboot“ in den Default-Runlevel**

```
# chmod 755 /etc/init.d/vnumlboot
# re-update add vnumlboot default
* vnumlboot added to runlevel default
```
Die Terminal-Ausgabe des Gentoo UML-Rechners wird über die Datei „/etc/inittab“ konfiguriert. Normalerweise öffnen sich sechs X-Terminals für jeden UML-Rechner wenn die Ausgabe nicht beim Start über die „con“-Optionen eingeschränkt wird.


Der Eintrag „tty0“ muss sich auch in der Datei „/etc/securetty“ befinden, was aber für gewöhnlich bereits der Fall ist.


```
# rc-update add sshd default
* sshd added to runlevel default
*rc-update complete
```

Die VNUML-Autoren empfehlen eine Veränderung der Start-Optionen des SSH-Servers um „sshd -u0“. Dabei wird die Länge des Feldes, das den Remote Host Namen in der „utmp“-Struktur enthält auf Null gesetzt. Dies vermeidet DNS-Anfragen des SSH-Servers die den SSH-Zugriff in VNUML Simulationen verzögern können. Um diese Modifizierung durchzuführen kann innerhalb der Datei „/etc/init.d/sshd“ in der start()-Methode hinter der aufgeführten Startsequenz „/usr/sbin/sshd“ einfach die Option -u0 gesetzt werden.

Nun sollten noch die nötigen Schlüssel für die verschiedenen SSH-Protokolle erzeugt werden. Dies geschieht wie folgt.

```
/ # /usr/bin/ssh-keygen -t rsa1 -b 1024 -f /etc/ssh/ssh_host_rsa_key -N "
/ # /usr/bin/ssh-keygen -d -f /etc/ssh/ssh_host_rsa_key -N "
```

Anleitung: 7 Erstellen eines VNUML-Rechners auf Basis von Gentoo-Linux


Im folgenden wird zuerst der ursprüngliche Quellcode in den einzelnen Dateien aufgezeigt und anschließend der modifizierte Quellcode, in dem die Veränderungen zusätzlich durch Fett-Schrift markiert sind.

In der Datei „/etc/init.d/modules“ innerhalb der Methode start():

```bash
if [[ /etc/modules.d -nt /etc/modules.conf ]] ; then
    ebegin "Calculating module dependencies"
    /sbin/modules-update
    eend ?? "Failed to calculate dependencies"
else
    einfo "Module dependencies are up-to-date"
fi
```

Listing 7.5.2: Modifizierter Code der Datei „/etc/init.d/modules“

Diese Stelle im Quellcode muss wie folgt um eine if-Anweisung erweitert werden:

```bash
if [[ /etc/modules.d -nt /etc/modules.conf ]] ; then
    ebegin "Calculating module dependencies"
    /sbin/modules-update
    eend ?? "Failed to calculate dependencies"
else
    einfo "Module dependencies are up-to-date"
fi
```

Listing 7.5.2: Modifizierter Code der Datei „/etc/init.d/modules“

In der Datei „/etc/init.d/localmount“ innerhalb der Methode start():

```bash
mount -at noproc,noshm,no${NET_FS_LIST// /,no} > /dev/null
```

Listing 7.6.1: Original Code der Datei „/etc/init.d/localmount“

Diese Stelle im Quellcode der Datei muss um die Option F, für das parallele Mounten, erweitert werden:

```bash
mount -aFf noproc,noshm,no${NET_FS_LIST// /,no} > /dev/null
```

Listing 7.6.2: Modifizierter Code der Datei „/etc/init.d/localmount“
Anleitung: 7 Erstellen eines VNUML-Rechners auf Basis von Gentoo-Linux

In der Datei „/etc/init.d/bootmisc“ innerhalb der Methode start():

```bash
if [[ -x /sbin/env-update.sh ]]; then
    ebegin "Updating environment"
    /sbin/env-update.sh > /dev/null
    eend 0
fi
```

Listing 7.7.1: Original Code der Datei „/etc/init.d/bootmisc“

Diese Stelle im Quellcode muss um eine if-Anweisung erweitert werden:

```bash
if [[ -x /sbin/env-update.sh ]]; then
    if [[ /etc/env.d -nt /etc/profile.env ]]; then
        ebegin "Updating environment"
        /sbin/env-update.sh > /dev/null
        eend 0
    else
        einfo "Environment up-to-date"
    fi
fi
```

Listing 7.7.2: Modifizierter Code der Datei „/etc/init.d/bootmisc“

In der Datei „/etc/conf.d/rc“:

```bash
RC_PARALLEL_STARTUP=no
```

Listing 7.8.1: Original Code der Datei „/etc/conf.d/rc“

Für das parallele Starten der Dienste beim Hochfahren muss die Variable auf „yes“ gesetzt werden:

```bash
RC_PARALLEL_STARTUP=yes
```

Listing 7.8.2: Modifizierter Code der Datei „/etc/conf.d/rc“

Um die Länge der Bootzeit noch weiter zu verkürzen können einige nicht benötigte Dienst-Programme aus den Runleveln entfernt werden. Dazu dient das bereits vorgestellte Programm „rc-update“. Mittels „rc-update“ können die Init-Scripte, die auf die Programme verweisen, angezeigt und aus den Runleveln entfernt werden. Über den Befehl „rc-update show“ können alle Init-Scripte und die zugehörigen Runlevel, aus denen sie referenziert werden, angezeigt werden.

```bash
# rc-update show
  bootmisc  | boot
  checkfs   | boot
  checkroot | boot
  [...]     |
  sshd      | default
  vnumlboot | default
```

Befehl 7.11: Anzeigen der Init-Scripte, die beim Hochfahren automatisch ausgeführt werden

Ein Teil dieser Init-Scripte kann aus den Runleveln entfernt werden, da ihre Funktion für die virtuelle Netzwerk-Umgebung nicht notwendig ist. Im folgenden sind fünf Init-Scripte aufgelistet, deren entfernen kein wirklicher Nachteil für die Funktion des Gentoo-Systems bedeutet:

```bash
# rc-update del consolefont
  boot
" " domainname
" " hostname
" " keymaps
" " local
" " local nonetwork
```

Befehl 7.12: Entfernen einiger nicht benötigter Init-Scripte aus deren Runlevel

Damit ist das Gentoo-Root-Dateisystem für den Einsatz in VNUML fertig konfiguriert. Für die vollständige Funktion fehlt allerdings noch der „devfs“-Daemon, der im Kapitel 7.2.3 installiert wird.

Listing 7.9: Deaktivieren der „rp_filter“ Variable über die Datei „/etc/sysctl.conf“


Listing 7.10: Konfiguration der zweiten Partition in „/etc/fstab“ für VNUML 1.6

Statt des Verzeichnisses „/mnt/vnuml/“ verlangt VNUML 1.6 ein eigenständiges Verzeichnis „/root/.ssh“, da in die zweite Partition aufgrund des ISO-Formats nun auch nicht mehr geschrieben werden kann.

Listing 7.11: Alternative Konfiguration für „udev“ über die „udev“-Regeldatei 50-udev.rules

Ansonsten müssen keine weiteren Veränderungen für die Verwendung des Root-Dateisystems mit VNUML 1.6 durchgeführt werden.


Listing 7.12: Konfiguration der zweiten Partition in „/etc/fstab“ für VNUML 1.6

Listing 7.13: Konfiguration des „.ssh“-Verzeichnisses für VNUML 1.6

Listing 7.11: Alternative Konfiguration für „udev“ über die „udev“-Regeldatei 50-udev.rules


7.2.3 Installation benötigter Software


Die folgenden Optionen von „emerge“ sind weitestgehend die wichtigsten:

- `emerge --sync` - aktualisiert den „Portage-Tree“ und die „ebuild“-Scripte
- `emerge „Softwarepaket“` - installiert ein Softwarepaket inklusive aller abhängigen Softwarepakete
- `emerge -s „Suchwort“` - Zum suchen nach Paketen, in deren Namen das Suchwort enthalten ist
- `emerge -S „Suchwort“` - Zum suchen nach Paketen, in deren Beschreibung das Suchwort ist
- `emerge -p „Softwarepaket“` - Installiert nichts, zeigt lediglich die von dem Softwarepaket abhängigen Pakete an, bzw. was installiert werden würde, wenn -p nicht angegeben wäre
- `emerge -pv „Softwarepaket“` - Installiert nichts und zeigt neben den Abhängigkeiten auch die möglichen USE-Flags der Softwarepakete an
- `emerge -O „Softwarepaket“` - versucht nur das angegebene Softwarepaket zu installieren, ohne die abhängigen Softwarepakete
- `emerge -f „Softwarepaket“` - lädt lediglich die Quelltexte des angegebenen und der abhängigen Softwarepakete aus dem Internet, ohne die Installation durchzuführen
- `emerge -u „Softwarepaket“` - aktualisiert die installierte Software inklusive der Software der abhängigen Pakete
- `emerge --info` - zeigt Informationen bzgl. der Installationsroutine an
- `emerge --unmerge „Softwarepaket“` - deinstalliert die Software des angegebenen Paketes


```
USE="+X snmp smux ssl ipv6"
```

Listing 7.12: globale Einstellung der USE-Flags über die Datei „/etc/make.conf“


```
GENTOO_MIRRORS="ftp://ftp.uni-koblenz.de/pub/mirrors/gentoo/"
```

Listing 7.13: Einstellen von Gentoo-Servern für den Software-Download


So finden sich in dieser Datei für das Softwarepaket „net-snmp“ die folgenden Einträge.

```
net-analyzer/net-snmp:diskio - Enable the use of diskio mibs
net-analyzer/net-snmp:elf - Enable the use of elf utils to check uptime on some systems
net-analyzer/net-snmp:mfd-rewrites - Use MFD rewrites of mib modules where available
net-analyzer/net-snmp:rpm - Enable the rpm snmp probing
net-analyzer/net-snmp:smux - Enable the smux MIBS module
```

Listing 7.14: USE-Flags zu net-snmp erläutert in der Datei „/usr/portage/profiles/use.local.desc“

Die hier aufgelisteten USE-Flags müssen aber nicht unbedingt für das aktuelle Softwarepaket gültig sein.


```
/# emerge -s net-snmp
* net-analyzer/net-snmp
  Latest version available: 5.2.1.2-r1
  Latest version installed: [ Not Installed ]
  Size of files: 3,779 kB
  Homepage: http://net-snmp.sourceforge.net/
  Description: Software for generating and retrieving SNMP data
  License: as-is BSD
```

Befehl 7.14: Suche nach der Software „Net-SNMP“


```
/# ACCEPT_KEYWORDS="~x86" emerge -s net-snmp
* net-analyzer/net-snmp
  Latest version available: 5.2.2-r3
  Latest version installed: [ Not Installed ]
  Size of files: 3,827 kB
  Homepage: http://net-snmp.sourceforge.net/
  Description: Software for generating and retrieving SNMP data
  License: as-is BSD
```

Befehl 7.15: Suche nach der aktuellen „Net-SNMP“-Software aus dem Test-Zweig


```
/# emerge devfsd
```

Befehl 7.16: Installation des „devfs“-Daeemon

Von nun an ist das UML-Root-Dateisystem einsatzfähig und kann, wie im Kapitel 7.2.4 beschrieben, in eine Image datei abgelegt und schließlich mit einem UML-Kernel zusammen, auch in VNUML, verwendet werden. Im folgenden werden noch einige Beispiele für die Installation von Softwareprogrammen aufgeführt, die Teilweise auch in dem UML-Root-Dateisystem der VNUML-Autoren zu finden sind.
Zuerst wird hier jedoch die Installation von SNMP zusammen mit Quagga und der SMUX-Unterstützung aufgezeigt.

Über den folgenden Befehl können die dafür benötigten Softwarepakete zusammen mit den möglichen USE-Flags angezeigt werden. Hier werden die Optionen der aktuellsten Version aus dem Test-Zweig angezeigt.

```
/ # ACCEPT_KEYWORDS="~x86" emerge -pv net-snmp quagga

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild N ] net-analyzer/net-snmp-5.2.2-r3  USE="ipv6 smux ssl tcpd -X -doc -elf -lm_sensors -mfd-rewrites -minimal -rpm" 3,827 kB
[ebuild N ] net-misc/quagga-0.99.4 USE="ipv6 pam snmp -bgpclassless -multipath -ospfapi -realms -tcp-zebra -tcpmd5" 2,173 kB

Total size of downloads: 6,001 kB
```

Befehl 7.17: Anzeigen der Installationsattribute für Net-SNMP und Quagga

Je nachdem wie viele Abhängigkeiten noch nicht erfüllt sind, kann die Ausgabe hier variieren. Die Software kann dann zum Beispiel mit den folgenden Einstellungen installiert werden.

```
/ # ACCEPT_KEYWORDS="~x86" USE="mfd-rewrites tcp-zebra tcpmd5 -pam" emerge net-snmp quagga
```

Befehl 7.18: Installation von Net-SNMP und Quagga

Einige USE-Flags wurden bereits über die Datei „make.conf“ gesetzt, so dass auf deren Angabe hier verzichtet werden kann.


```
host # Xnest -ac -geometry 1024x768+0+0 :1
```

Befehl 7.19: Einrichten eines „nested X-Servers“ auf dem Host-System

Hierbei öffnet sich ein Fenster von 1024x768 Pixeln Größe an der Position 0;0 auf dem Desktop des Host-Systems.


```
uml1 # export DISPLAY=“192.168.0.1:1“
```

Befehl 7.20: Einrichten des UML-Rechners für die grafische Ausgabe und Start von „tkmib“


```
Befehl 7.21: Installation einiger Diagnose- und Hilfswerkzeuge
```


```
/ # emerge -O iptables ppp
```

Befehl 7.22: Installation von „iptables“ und „ppp“


```
/ # ebuild /usr/portage/net-analyzer/net-snmp/net-snmp-5.2.2-r3.ebuild unpack
```

Befehl 7.23: Herunterladen und Entpacken der Software Net-SNMP über das Gentoo-Programm „ebuild“

Somit kann die Software auch manuell aus dem Quellcode kompiliert und installiert werden, um Einstellungen zu konfigurieren, die auch „Portage“ nicht bietet. Wenn die Software zuvor über „Portage“ ohne Probleme automatisch installiert wurde, sollte eine manuelle Installation ebenfalls ohne Probleme durchführbar sein.

Ist die benötigte Software installiert, kann die „chroot“-Umgebung über den Befehl „exit“ wieder verlassen werden.

### 7.2.4 Erstellen der Imagedatei

Hier wird nun die Imagedatei erstellt, in die das zuvor konfigurierte UML-Root-Dateisystem hinein kopiert werden muss, um in VNUML verwendet werden zu können. Das System hätte auch direkt in der Imagedatei erstellt werden können. Die Imagedatei hat jedoch eine feste Größe und es ist Anfangs unbekannt wie groß das UML-Root-Dateisystem letztlich wird, weswegen die Imagedatei höchstwahrscheinlich entweder viel zu groß oder zu klein erstellt worden wäre.

Nachdem verlassen der „chroot“-Umgebung sollten zuerst die eingebundenen Verzeichnisse gelöst werden.

```
host # umount gentoo_backup/usr/portage
host # umount gentoo_backup/proc/
```

Befehl 7.24: Löschen der eingebundenen Verzeichnisse aus dem Gentoo-System
Anleitung: Erstellen eines VNUML-Rechners auf Basis von Gentoo-Linux

Sofern sich kein benötigter Software-Quellcode im Verzeichnis „/var/tmp/portage/“ befindet, kann der Inhalt des Verzeichnisses gelöscht werden um weiteren Speicherplatz zu sparen.

Was nun noch fehlt sind die Module des Kernels, die im Verzeichnis „gentoo_backup/lib/modules/“ abgelegt sein müssen. Dort müssen sie sich in einem Verzeichnis befinden, dessen Name der Bezeichnung des verwendeten UML-Kernels entspricht.

Die VNUML-Autoren liefern ihre Kernel-Module zusammen mit dem Kernel, gepackt in einem Tarball, so dass auch der UML-Kernel der VNUML-Autoren verwendet werden kann, nachdem die Kernel-Module innerhalb des „/lib/modules/“ Verzeichnisses entpackt wurden.


Die Größe der Imagedatei sollte die Größe des belegten Speicherplatzes um einiges überschreiten, damit es im laufenden Betrieb nicht zu Fehlern kommt. Mit dem Programm „dd“ (Disk Dump) wird die Imagedatei erstellt und anschließend mit dem „ext2“-Dateisystem formatiert.


Nun kann die Imagedatei in die Verzeichnis-Struktur des Hosts eingebunden und die Dateien des UML-Root-Dateisystems aus dem Backup-Verzeichnis hinüber kopiert werden, was eine längere Zeit in Anspruch nimmt.

Erstellen eines UML-Kernels


Nachdem entpacken des Tarballs kann mit der Konfiguration des Kernels direkt begonnen werden. Wichtig dabei ist, dass allen Kernel-Befehlen die Option „ARCH=um“ angehängt wird.


Nachdem entpacken des Tarballs kann mit der Konfiguration des Kernels direkt begonnen werden. Wichtig dabei ist, dass allen Kernel-Befehlen die Option „ARCH=um“ angehängt wird.

7.4 Starten des UML-Rechners

Mit dem UML-Kernel und dem UML-Root-Dateisystem kann nun schon die Funktion des UML-Rechners, auch außerhalb von VNUML, getestet werden.

Mit den folgenden Befehlen wird eine virtuelle Netzwerkschnittstelle im Host-System erstellt. Anschließend wird ein UML-Rechner gestartet, der diese Netzwerkschnittstelle verwenden kann.

```
host # tunctl -t tap0 -f /dev/net/tun
host # ./vmlinux ubd0=cow_fs,gentoo_root eth0=tuntap,tap0,fd:fe:0:0:1,192.168.0.254 umid=gentooUML
```

Befehl 7.34: Starten eines einzelnen UML-Rechners

Beim Start kommt es zwar zu einer Fehlermeldung aufgrund der fehlenden zweiten Partition. Dies beeinflusst die Funktion des UML-Rechners allerdings nicht.

Informationen zur weiteren Vorgehensweise finden sich im Kapitel 5.6, in den Befehlen 5.15 und 5.16.


```
host # uml_mconsole gentooUML
(gen_con) config ubd2=portage.img
OK
```

Befehl 7.35: Anlegen einer Imagedatei als Gerätedatei im laufenden UML-Rechner

Anschließend kann die neue Gerätedatei über den Befehl „mount“, innerhalb des UML-Rechners in dessen Verzeichnisstruktur eingebunden werden.

```
uml # mount /dev/ubd/2 /usr/portage
```

Befehl 7.36: Einbinden der neuen Gerätedatei in die Verzeichnisstruktur des UML-Rechner


7.5 Zusammenfassung

Abbildungsverzeichnis
Abb. 3.1: Die Funktionsbereiche des Netzwerkmanagements 9
Abb. 4.1: Das SNMP Manager/Agenten Modell 12
Abb. 4.3: Standardisierungsprozess der IETF [9] intro-standards.pdf 14
Abb. 4.4: Netzwerkmanagement-Architektur des SNMP-Modells [7] 17
Abb. 4.5: SNMP-Architektur in TCP/IP-Netzen 18
Abb. 4.6: Architektur des SNMPv3-Managers [9] snmpv3.pdf 19
Abb. 4.7: Architektur des SNMPv3-Agenten [9] snmpv3.pdf 20
Abb. 4.8: Ausschnitt des Objekt-Registrierungsbaumes der ISO 22
Abb. 4.9: Struktur mit BER kodierter Daten 24
Abb. 4.11: Beispiel für die Kodierung eines komplexeren ASN.1 Datensatzes mit BER 25
Abb. 4.12: Baum-Ausschnitt der Gruppe „interfaces“ 31
Abb. 4.15: Ablauf einer GetBulkRequest Kommunikation [9] snmpv2.pdf 34
Abb. 4.16: Ablauf einer SetRequest Kommunikation [9] snmpv2.pdf 34
Abb. 4.18: Ablauf der InformRequest Meldung [9] snmpv2.pdf 35
Abb. 4.19: Aufbau der SNMP-PDU 36
Abb. 4.20: Aufbau der SNMPv1 und -v2c Nachrichten 36
Abb. 4.23: Master/Sub-Agenten Architektur 41
Abb. 5.1: vnumlGUI, ein grafisches Benutzerinterface zu VNUML 46
Abb. 5.2: Topologie des Netzwerk-Szenarios „SimpleNet“ 47
Abb. 5.3: Komplette virtuelle Netzwerktopologie der VNUML Simulation des „SimpleNet“-Szenarios 50
Abb. 5.4: allgemeiner Aufbau eines Linux-Betriebssystems 52
Abb. 5.5: Der Copy On Write (COW) Mechanismus 55
Abb. 5.6: Mehrere COW-Imagedateien über einer Basis-Imagedatei 55
Abb. 5.7: Manuell erstellter Teil des „SimpleNet“-Szenarios 56
Abb. 5.8: Verteilung der Prozesse des virtuellen Netzerwerks im Host-System 59
Abb. 5.9: Netz-Topologie der UML-Internet-Verbindung 61
Abb. 5.10: Einige Quagga Routing Dienste 63
Abb. 6.1: Möglichkeiten des Einsatzes:
SNMP als Erweiterung des VNUML-Managements oder in isolierter Simulations-Umgebung 72
Abb. 6.2: MIB-Modul „studienarbeit“ im ISO-Registrierungsbaum 93
Abb. 6.3: Ablauf der Set-Befehl Verarbeitung 96

Tabellenverzeichnis
Tabelle 4.1: Einige wichtige RFC-Dokumente zu SNMP 16
Tabelle 4.2: Lexikalische Konventionen von SMI 21
Tabelle 4.3: Basis-Datentypen, die SNMP verwenden kann 21
Tabelle 4.4: BER-Kodierungszahlen einiger universeller Datentypen 24
Tabelle 4.5: Einige SMIV2 Datentypen für SNMP Management-Objekte 26
Tabelle 4.6: Einige SMIV2 Textkonventionen (Entnommen RFC 2579) 27
Tabelle 4.7: Als Liste dargestellter Ausschnitt der Gruppe „interfaces“ 31
Tabelle 4.8: Einige Gruppen der MIB-II 32
Tabelle 4.9: Die in SNMPv1 definierten Trap-Nachrichten 35
Tabelle 6.1: Vergleich des Ressourcenverbrauchs der einzelnen SNMP-Versionen 104
Listingverzeichnis

Listing 4.1: Definition eines Integers in ASN.1 21
Listing 4.2: Definition eines wertebeschränkten Integer-Datentyps in ASN.1 21
Listing 4.3: Tagging von Datentypen 23
Listing 4.4: Definition der Textkonvention „RunState“ 27
Listing 4.5: Textuelles Gerüst der MIB-Dateien 28
Listing 4.6: Definition eines Objektes in SMIv2 28
Listing 4.7: Auszug aus der SNMPv2-MIB (Entnommen RFC 3418) 29
Listing 4.8: Auszug der Definition der Tabelle „ifTable“ (Entnommen RFC 2863 - The Interfaces Group) 30
Listing 4.9: Definition eines Notification-Objektes für SNMP-Meldungen 31
Listing 5.1.1: Inhalt der Datei SimpleNet.xml, Teil 1 48
Listing 5.1.2: Inhalt der Datei SimpleNet.xml, Teil 2 48
Listing 5.1.3: Inhalt der Datei SimpleNet.xml, Teil 3 49
Listing 5.2: Inhalt der Datei „/etc/fstab“ des UML-Root-Dateisystems bei Verwendung von „devfs“ 54
Listing 5.3: Inhalt der „zebra“-Konfigurationsdatei „zebra.conf“ 66
Listing 5.4: Inhalt der „ripd“-Konfigurationsdatei „ripd.conf“ 67
Listing 5.5: Inhalt der Datei „snmpd.log“ nach Zustandekommen einer SMUX-Verbindung 68
Listing 6.1: Eintrag einer Quelltext-Downloadquelle für die Debian APT-Tools in die Datei „sources.list“ 72
Listing 6.2: Möglichkeiten der Objekt-Bezeichnung des Managers 76
Listing 6.3: Bash-Script „snmpQuery.sh“ mit eingebundenen Net-SNMP Anwendungen des Managers 79
Listing 6.4.1: Konfiguration der Zugriffsschrechte in der Datei „snmptrapd.conf“ des Trap-Server 82
Listing 6.4.2: Konfiguration des Trap-Servers für das Verarbeiten von Meldungen 82
Listing 6.5.1: Konfiguration für den SNMPv1 / v2c Zugriff in der Datei „snmp.conf“ 83
Listing 6.5.2: Konfiguration für den SNMPv3 Benutzerzugriff in der Datei „snmp.conf“ 83
Listing 6.5.3: Konfiguration von VACM Security-Namen in der Datei „snmp.conf“ 84
Listing 6.5.4: Konfiguration von VACM Gruppen in der Datei „snmp.conf“ 84
Listing 6.5.5: Konfiguration von VACM Sichten in der Datei „snmp.conf“ 84
Listing 6.5.6: Konfiguration von VACM Access Policies in der Datei „snmp.conf“ 85
Listing 6.5.7: Konfiguration von Systeminformationen des Agenten in der Datei „snmp.conf“ 85
Listing 6.5.8: Konfiguration der Ziele für SNMP-Meldungen in der Datei „snmp.conf“ 86
Listing 6.5.9: Konfiguration der Übergewicht der Prozessinstanzen in der Datei „snmp.conf“ 86
Listing 6.5.10: Konfiguration der Überwachung der Speicherkapazität und Dateigröße 87
Listing 6.5.11: Konfiguration für das Senden von Standard-Ereignismeldungen 87
Listing 6.5.12: Konfiguration für das Senden einer Meldung bei fehlgeschlagenen Authentifizierung 87
Listing 6.5.13: Konfiguration für das Senden von Ereignismeldungen für fehlende Prozessinstanzen 87
Listing 6.5.14: Konfiguration der Einbindung externer Programme über „exec“ 88
Listing 6.5.15: Konfiguration von extern verwalteten MIB-Bibliotheken über „pass“ und „pass_persist“ 89
Listing 6.5.16: Konfiguration von Quagga SMUX-Peers in der Datei „snmp.conf“ 90
Listing 6.5.17: Konfiguration des AgentX Sub-Agenten in der Datei „snmp.conf“ 91
Listing 6.6: Beispiel eines Script-Programms „testprogie.sh“ zum Einbinden in den Net-SNMP Agenten 88
Listing 6.7: Beispiel für ein durch „pass“ angesteuertes Bash-Script „svtyshQuery.sh“ 89
Listing 6.8: OID der Quagga beiligenden MIBs (MIB-Module) 90
Listing 6.9: Inhalt der MIB-Datei „STUDIENARBEIT-MIB.txt“ 93
Listing 6.10.1: Inhalt der Datei „studienarbeit.c“, Teil 1 95
Listing 6.10.2: Inhalt der Datei „studienarbeit.c“, Teil 2 95
Listing 6.10.3: Inhalt der Datei „studienarbeit.c“, Teil 3 95
Listing 6.10.4: Inhalt der Datei „studienarbeit.c“, Teil 4 96
Listing 6.10.5: Inhalt der Datei „studienarbeit.c“, Teil 5 97
Listing 6.11: Inhalt des Programms „ifTraffic.sh“ 101
Listing 6.12: Inhalt der Datei „ifTrafficAgent_eth1.csv“ 101
Listing 6.13: Inhalt der Script-Datei „zebraTest.sh“ 102
Listing 6.14: Definition des Notification-Objects „noZebra“ 102
Listing 6.15: Inhalt der Log-Datei des Trap-Servers nach Erhalt der „noZebra“ Meldung 103
Listing 6.16: Konfiguration der „snmptrapd.conf“ für den Empfang der Ereignismeldung „noZebra“ 104

- 122 -
Abbildungs-, Tabellen-, Listing-, Befehlsverzeichnis

Listing 6.17: Inhalt des Bash-Programms „zebra.sh“

Listing 6.18: Inhalt der Datei „zebraOutput.txt“

Listing 7.1: Konfiguration der Datei „/etc/fstab“

Listing 7.2: Das Init-Script „vnumlboot“ für die Referenz aus dem Runlevel auf „umlboot“

Listing 7.3: Konfiguration der Login-Terminals über die Datei „/etc/ssh/sshd_config“

Listing 7.5.1: Original Code der Datei „/etc/init.d/modules“

Listing 7.5.2: Modifizierter Code der Datei „/etc/init.d/modules“

Listing 7.6.1: Original Code der Datei „/etc/init.d/localmount“

Listing 7.6.2: Modifizierter Code der Datei „/etc/init.d/localmount“

Listing 7.7.1: Original Code der Datei „/etc/init.d/bootsmisc“

Listing 7.7.2: Modifizierter Code der Datei „/etc/init.d/bootsmisc“

Listing 7.8.1: Original Code der Datei „/etc/conf.d/rc“

Listing 7.8.2: Modifizierter Code der Datei „/etc/conf.d/rc“

Listing 7.9: Deaktivieren der „rp_filter“ Variable über die Datei „/etc/sysctl.conf“

Listing 7.10: Konfiguration der zweiten Partition in „/etc/fstab“ für VNUML 1.6

Listing 7.11: Alternative Konfiguration für „udev“ über die „udev“-Regeldatei 50-udev.rules

Listing 7.12: globale Einstellung der USE-Flags über die Datei „/etc/make.conf“

Listing 7.13: Einstellen von Gentoo-Servern für den Software-Download

Listing 7.14: USE-Flags zu net-snmp erläutert in der Datei „/usr/portage/profiles/use.local.desc“

Befehlsverzeichnis

Befehl 4.1: SNMPv1 Anfrage, ausgeführt mit Net-SNMP 38

Befehl 5.1: Starten der Netzwerk-Simulation „SimpleNet“ 50

Befehl 5.2: Kopieren der benötigten Konfigurationsdateien vom Host auf die UML-Rechner 50

Befehl 5.3: Start der Dienste auf den UML-Rechnern 50

Befehl 5.4: Ausschnitt der Ausgabe des Simulations-Kommandos „mgmt“ 50

Befehl 5.5: Beenden einer Netzwerk-Simulation 51

Befehl 5.6: Abbruch einer Netzwerk-Simulation und Löschen aller Backup-Dateien 51

Befehl 5.7: Routenverfolgung von „uml1“ zu „uml3“, vom Host via „ssh“ aus initiiert 51

Befehl 5.8: Kopieren der Datei „ripd.conf“ auf den UML-Rechner „uml2“ mittels „scp“ 51

Befehl 5.9: Starten eines UML-Rechners 54

Befehl 5.10: Verschmelzen einer Basis-Imagedatei mit der zugehörigen COW-Imagedatei 55

Befehl 5.11.1 : Aufbau des virtuellen Netzes im Host 56

Befehl 5.11.2 : Konfiguration der Management-Netzwerkschnittstellen des Hosts 56

Befehl 5.11.3 : Hinzufügen der Netzwerkschnittstellen zur Bridge 57

Befehl 5.11.4: Erstellen der Imagedatei für das Konfigurations-Script des UML-Rechners „uml1“ 57

Befehl 5.11.5: Kopieren des SSH-Schlüssels 57

Befehl 5.11.6: Erstellen der Script-Datei „umlboot“ für die Konfiguration des UML-Rechners „uml1“ 57

Befehl 5.11.7: Erstellen der Image- und Script-Datei für die Konfiguration des UML-Rechners „uml2“ 58

Befehl 5.12 : Verwendung der UML-Management-Konsole „uml_mconsole“ 58

Befehl 5.13: Vorbereitung der Imagedatei für die Software Installation 60

Befehl 5.14: Verlassen der „chroot“-Umgebung und lösen der Imagedatei 60

Befehl 5.15: Einrichten eines NAT-Routers auf dem Host für einen UML-Rechner 61

Befehl 5.16: Einrichten einer Internetverbindung für den UML-Rechner „uml1“ 61

Befehl 5.17: Vergrößern einer vorhandenen Imagedatei 62

Befehl 5.18: Erstellen einer neuen Imagedatei für UML-Rechner 63

Befehl 5.19: Herunterladen und Entpacken des Quagga-Quellcodes 64

Befehl 5.20: Manuelle Installation von Quagga 64

Befehl 5.21: Lokalisierung der Quagga Routing-Dienstprogramme 64

Befehl 5.22: Überprüfen der Referenz auf die Net-SNMP-Bibliothek 64

Befehl 5.23: Aufspüren der MIB-Dateien der Quagga Routing-Dienste 65

Befehl 5.24: Start des Quagga Kernel-Routing-Table-Managers „zebra“ 65
Abbildungs-, Tabellen-, Listing-, Befehlsverzeichnis

Befehl 5.25: Start des Quagga RIP-Routers „ripd“
Befehl 5.26: Einloggen in die „zebra“-Terminal-Shell via „Telnet“
Befehl 5.27: Wechseln in den „Enable Mode“ von „zebra“
Befehl 5.28: Auflistung aller gültigen Anweisungen innerhalb „zebra“
Befehl 5.29: Einloggen in die „ripd“-Terminal-Shell
Befehl 5.30: Einloggen in die „vtysh“-Terminal-Shell
Befehl 5.31: Anzeigen von RIP Routing-Informationen über „vtysh“ in der Konsole
Befehl 5.32: SNMP-Abfrage von Managementinformationen des RIP-Routers

Befehl 6.1: Download des Net-SNMP Quellcodes mittels Debians „apt-get“
Befehl 6.2: Installation der Software, die Net-SNMP für seine Funktion benötigt
Befehl 6.3: Konfiguration für das Kompilieren von Net-SNMP
Befehl 6.4: Installation von Net-SNMP
Befehl 6.5: Allgemeines Verwendungs-Schema der Kommandozeilen-Programme des Managers
Befehl 6.6: Get-Anfrage mit Net-SNMPs „snmpget“
Befehl 6.7: SNMPv3 Get-Anfrage mit Net-SNMPs „snmpget“
Befehl 6.8: GetNext-Anfrage mit Net-SNMPs „snmpgetnext“
Befehl 6.9: GetBulk Anfrage mit Net-SNMPs „snmpbulkget“
Befehl 6.10: Set-Aufforderung mit Net-SNMPs „snmpset“
Befehl 6.11: Auflistung aller verfügbaren Managementinformationen der UDP-MIB mit „snmpwalk“
Befehl 6.12: Abfrage von Objekt-Definitionen mittels „snmptable“
Befehl 6.14: Starten des Trap-Servers „snmptrapd“ mit Konsolausgabe
Befehl 6.15: Starten des Trap-Servers „snmptrapd“ mit Konfigurations- und Log-Datei
Befehl 6.16: Erstellen einer Konfigurationsdatei für den Agenten mittels „snmpconf“
Befehl 6.17: Starten des Net-SNMP Agenten
Befehl 6.18: Einrichten eines SNMPv3-Benutzers mit dem Programm „net-snmp-config“
Befehl 6.19: „snmpwalk“-Ausgabe der VACM-Community „allpublic“
Befehl 6.20: Abfrage der Prozessüberwachungstabelle „ucdavis.prTable“
Befehl 6.21: Ausgabe des Bash-Scripts „testprogie.sh“
Befehl 6.22: Anzeigen der Tabelle der durch „exec“ ausführbaren Programme auf dem Agenten
Befehl 6.23: SNMP-Abfrage der durch das Script „vtyshQuery.sh“ verwalteten Objekte
Befehl 6.24: Starten des Net-SNMP Agenten als AgentX Sub-Agent
Befehl 6.25: Versenden von SNMP-Meldungen
Befehl 6.26: Kompilieren eines AgentX Sub-Agenten für Net-SNMP
Befehl 6.27: Eingliederung der „STUDIENARBEIT-MIB“ in die Net-SNMP MIB-Umgebung
Befehl 6.28: Anzeigen der neuen Objekte mit „snmptranslate“
Befehl 6.29: Erstellen der C-Dateien auf Basis der STUDIENARBEIT-MIB
Befehl 6.30: Kopieren der „STUDIENARBEIT-Objekte“ direkt in den Net-SNMP Agenten
Befehl 6.31: Kompilieren der „STUDIENARBEIT-Objekte“ in einen AgentX Sub-Agenten
Befehl 6.32: Starten des Sub-Agenten studieProg
Befehl 6.34: Überprüfen der Netzwerk-Routen mit dem Befehl „route“
Befehl 6.35: Übergeben einer neuen IP-Route via SNMP über das Objekt „superUserCommand“
Befehl 6.36: Auslesen der Routing-Tabelle des Agenten via SNMP
Befehl 6.37: Konfiguration der Netzwerkschnittstellen mit dem Befehl „sysctl“
Befehl 6.38: Übermittlung des „sysctl“ Befehls an den Agenten via SNMP
Befehl 6.39: Abfrage IPv4-Weiterleitung über das „ipv4forwarding“ Objekt
Befehl 6.40: Setzen der IPv4-Weiterleitung via SNMP über die numerischen Werte
Befehl 6.41: Start des Bash-Programms „ifTraffic.sh“
Befehl 6.42: „snmptranslate“ über „studienarbeit“
Befehl 7.1: Download des „Stage3“- und des „Portage“-Tarballs
Befehl 7.2: Entpacken des „Stage3“-Tarballs
Befehl 7.3: Entpacken des „Portage“-Tarballs
Befehl 7.4: Kopieren der Datei „resolv.conf“ für den Zugriff auf Internet-Adressen
Befehl 7.5: Wechsel in die „chroot“-Umgebung in das Gentoo-Linux-System
Befehl 7.6: Setzen eines Passwortes für den UML-Rechner Benutzer „root“

Befehl 7.7: Erstellen des „Mountpunktes“ für die zweite Partition und der SSH-Verlinkung

Befehl 7.8: Setzen des Init-Scripts „vnumlboot“ in den Default-Runlevel

Befehl 7.9: Setzen des Init-Scripts „sshd“ in den Default-Runlevel

Befehl 7.10: Erzeugen der lokalen SSH-Schlüssel für den UML-Rechner

Befehl 7.11: Anzeigen der Init-Scripte, die beim Hochfahren automatisch ausgeführt werden

Befehl 7.12: Entfernen einiger nicht benötigter Init-Scripte aus den Runlevels

Befehl 7.13: Konfiguration des „.ssh“-Verzeichnisses für VNUMUL 1.6

Befehl 7.14: Suche nach dem Softwarepaket „Net-SNMP“

Befehl 7.15: Suche nach der aktuellen „Net-SNMP“-Software aus dem Test-Zweig

Befehl 7.16: Installation des „devfs“-Daemon

Befehl 7.17: Anzeigen der Installationsattribute für Net-SNMP und Quagga

Befehl 7.18: Installation von Net-SNMP und Quagga

Befehl 7.19: Einrichten eines „nested X-Servers“ auf dem Host-System

Befehl 7.20: Einrichten des UML-Rechners für die grafische Ausgabe und Start von „tkmib“

Befehl 7.21: Installation einiger Diagnose- und Hilfswerkzeuge

Befehl 7.22: Installation von „iptables“ und „ppp“

Befehl 7.23: Herunterladen und Entpacken der Software Net-SNMP über das Gentoo-Programm „ebuild“

Befehl 7.24: Löschen der eingebundenen Verzeichnisse aus dem Gentoo-System

Befehl 7.25: Bestimmen der benötigten Größe für die Imagedatei

Befehl 7.26: Erstellen und Formatieren der Imagedatei

Befehl 7.27: Kopieren der UML-Root-Dateisystem-Daten in die Imagedatei

Befehl 7.28: Abschalten des Konsistenzchecks des UML-Root-Dateisystems

Befehl 7.29: Herunterladen und Entpacken des Linux-Kernel-Quellcodes

Befehl 7.30: Übernehmen der VNUMUL-Kernel Konfigurationseinstellungen

Befehl 7.31: Konfigurieren des UML-Kernels

Befehl 7.32: Kompilieren des UML-Kernels

Befehl 7.33: Installation der UML-Kernel-Module

Befehl 7.34: Starten eines einzelnen UML-Rechners

Befehl 7.35: Anlegen einer Imagedatei als Gerätedatei im laufenden UML-Rechner

Befehl 7.36: Einbinden der neuen Gerätedatei in die Verzeichnisstruktur des UML-Rechner
Literaturverzeichnis