FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Portierung der DPI-Subagentenschnittstelle auf den UCD-SNMP-Agenten

Martin Friedrich

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Stephen Heilbronner
Abgabedatum:	15.12.1998

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Internet-Management

Das Management von Netzen und Systemen in der Internetwelt verwendet das standardisierte Simple Network Management Protocol (SNMP) [Case 90] [Case 93] [HA 93] [Halsall 95], welches auf dem Client-Server-Prinzip basiert. Der Client wird als Manager und der Server als Agent bezeichnet (Abbildung 1.1). Ein Agent verwal-

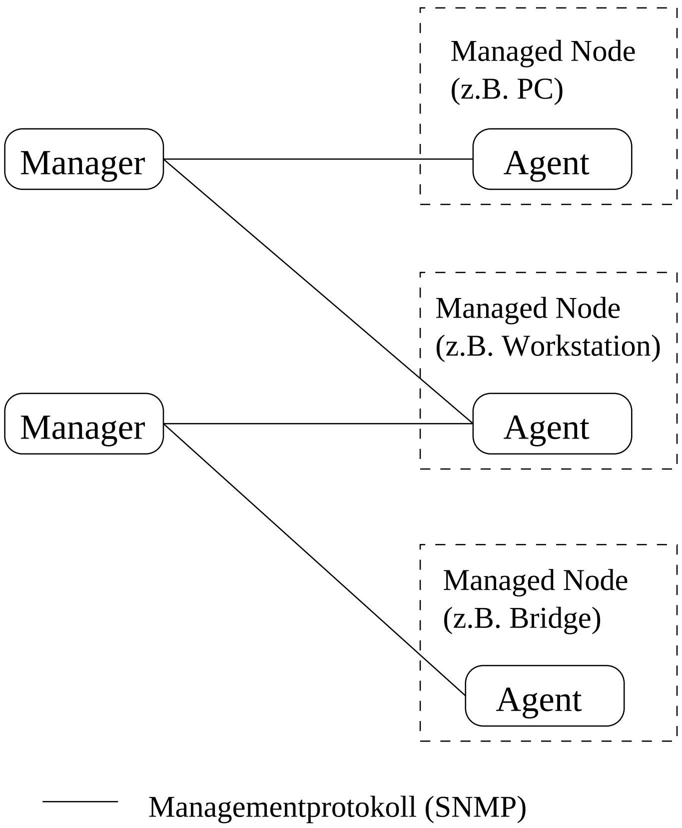

Abbildung 1.1: Internet-Organisationsmodell

tet Managementinformation, die sogenannte Management Information Base (MIB). Beispiele für Managementinformation sind:

- Anzahl der empfangenen IP-Pakete

- die Routingtabelle einer Workstation
- das Betriebssystem einer Workstation
- Konfigurationsparameter von Softwaresystemen

Der Manager kann mit dem SNMP-Protokoll Managementinformation vom Agenten lesen oder schreiben. Mit seinen Anfragen muß er natürlich die von ihm gewünschte Information eindeutig identifizieren. Für diesen Zweck existiert im Internet-Management der Internet-Registrierungsbaum[1] (Abbildung 1.2). Die Blätter dieses

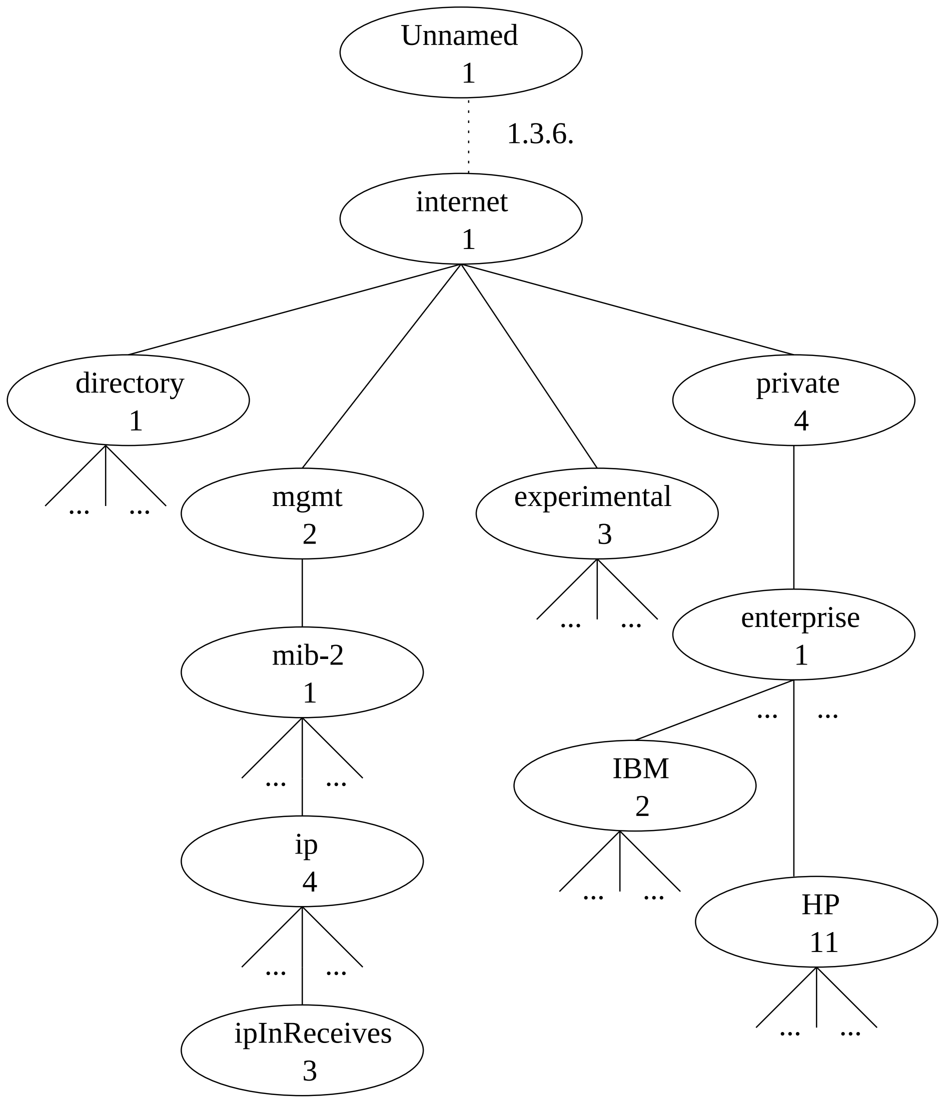

Abbildung 1.2: Internet-Registrierungsbaum

Baumes enthalten die Managementinformation in den MIB-Variablen. Die Knoten hingegen dienen der Strukurierung der MIB. Will man nun eine MIB-Variable identifizieren, so gibt man den Weg im Baum bis zu diesem Blatt an. Mit 1.3.6.1.2.1.4.3.0 kann man zum Beispiel die eingegangenen IP-Pakete (ipInReceives) abfragen. Die angehängte 0 verlangt die SNMP-Protokollspezifikation.

[1] auch als Internet-MIB bezeichnet

1.2 Probleme dieser Organisation

Bei Verwendung der Architektur aus Abbildung 1.1 für das Management ergibt sich diese Folgerung, wenn man weitere MIB-Variablen unterstützen will:

- Man schreibt einen neuen Agent
- oder man erweitert einen bereits bestehenden Agenten.

Der erste Ansatz stellt einen relativ hohen Programmieraufwand dar, wobei der zweite dafür gute Kenntnisse des Quellcodes voraussetzt. Zudem entsteht ein monolithischer Agent, dessen Wartung und Aktualisierung mit steigender Funktionalität immer komplexer wird. Oft ist aber eine Änderung des Agenten nicht möglich, da sein Quellcode nicht vorhanden ist, was bei kommerziellen Produkten die Regel ist.

1.3 Verbesserte Architektur durch das SNMP-DPI-Protokoll

IBM entwickelte nun zur Beseitigung dieser Probleme das SNMP-DPI-Protokoll (SNMP Distributed Program Interface) [Carpenter 91] [Wijnen 94]. Ziel dieser Entwicklung war ein Agent[2], der dynamisch um Funktionalität erweitert werden kann. Neue Aufgaben können nun unabhängig von diesem Hauptagenten in sogenannten Subagenten implementiert werden (Abbildung 1.3). Diese Subagenten können dann den Teilbaum der Internet-MIB, für die sie zuständig sein wollen, beim Hauptagenten registrieren lassen. Dieser delegiert dann unsichtbar für den Manager entsprechende Anforderungen an den Subagenten.

Dieser Lösungsansatz stellt nur einen von vielen denkbaren dar. Weitere wären:

- Shared Memory
- Shared Libraries

Bei Verwendung von Shared Memory erfolgt die Prozeßkommunikation von Hauptagent und Subagent nicht mit einem Protokoll, sondern über einen gemeinsam genutzten Speicher.

Anders sieht es hier bei der Lösung mit Shared Libraries aus, hier sind der Hauptagent und die Subagenten in einem Programm verwirklicht. Die Subagenten werden als Shared Libraries realisiert, die vom Hauptagenten nachgeladen werden.

1.4 Aufgabenstellung

Im praktischen Teil dieser Arbeit hatte der Praktikant die Aufgabe, einen Agenten zu einem Hauptagenten zu erweitern und diesen zu testen.

1.5 Konventionen

Es sollen folgende Schreibkonventionen gelten:

- Für Datei- und Directory-Namen wir die Schrift Sans Serif verwendet.
- Auszüge aus dem Quellcode werden mit der Schrift `Typewriter` kleiner dargestellt.

[2] auch als Hauptagent bezeichnet

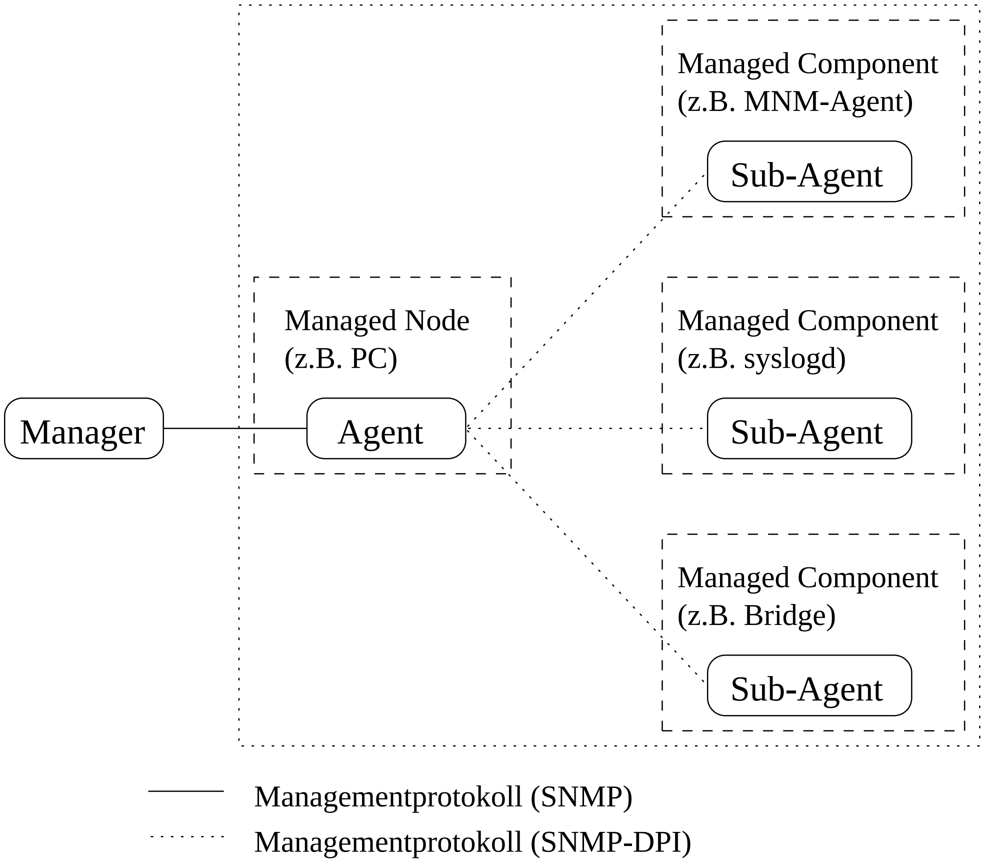

Abbildung 1.3: Internet-Architekturmodell mit dem SNMP-DPI-Protokoll

- Die Angabe von Kommandos folgt auch in `Typewriter`, aber in Normalgröße[3].
- Namen für Protokollpakete werden mit GROßEN **`fetten`** Buchstaben geschrieben.
- Funktionsnamen werden *kursiv* dargestellt.

[3]Falls dies den Zeilenrand überschreiten sollte, erfolgt auch diese Angabe verkleinert.

Kapitel 2

SNMP und SNMP-DPI

2.1 Das SNMP-Protokoll

Das Simple Network Management Protocol (SNMP) ist das Protokoll, daß für die Kommunikation von Manager und Agent verwendet wird [Case 90] [Case 93]. Es stellt fünf Protokollinteraktionen zur Verfügung:

1. **SNMP-GET**: Der Manager hat mit dieser Operation lesenden Zugriff auf eine MIB-Variable.
2. **SNMP-GETNEXT**: Der Manager erfragt mit dieser Operation einen Objektidentifikator und dessen Wert, der lexikographisch hinter dem im **SNMP-GETNEXT** angegebenen Objektidentifikator folgt.
3. **SNMP-GETBULK**[1]: Ähnlich wie **SNMP-GETNEXT**, jedoch können damit mehrer nachfolgende Objektidentifikatoren erfragt werden.
4. **SNMP-SET**: Der Manager kann hiermit MIB-Variablen Werte zuweisen.
5. **SNMP-TRAP**: Hiermit hat der Agent die Möglichkeit asynchrone Ereignisse dem Manager mitzuteilen.

2.2 Das SNMP-DPI-Protokoll

Die SNMP-DPI-Protokollpakete (SNMP Distributed Program Interface) kann man aufgrund ihrer Aufgaben in drei Kategorien einteilen [Carpenter 91] [Wijnen 94]:

1. Pakete für die Registrierung und Deregistrierung
2. Pakete, die denen der SNMP-Pakete entsprechen
3. Pakete, für die Implementierung eines Transaktionskonzepts

Die erste Aufgabe einer SNMP-DPI-Kommunikation zwischen Subagent und Hauptagent ist die Registrierung des Teilbaumes, für den der Subagent die Verantwortung übernehmen will. Die Registrierung läuft nun folgendermaßen ab:

1. Der Subagent schickt ein **SNMP-GET** an den Hauptagenten, in dem er die MIB-Variable 1.3.6.1.4.1.2.2.1.1.1.0 oder 1.3.6.1.4.1.2.2.1.1.2.0 (beide zusammen werden als DPI20-MIB bezeichnet) abfragt. Diese Variablen beinhalten die Portadressen für TCP und UDP, an denen der Hauptagent ankommende Registrierungswünsche entgegen nimmt.

[1] erst ab SNMP-Version 2.0

2. Nun kann der Subagent an diesen Port sein **DPI-OPEN** senden, daß dem Hauptagenten die Wurzel des Teilbaums mitteilt, für welche der Subagent zuständig sein möchte.

3. Mit **DPI-REGISTER** registriert der Subagent alle MIB-Variablen, die er bereitstellt.

Die Deregistrierung verläuft deutlich einfacher, und wird daher nicht näher beschrieben.

Ist der Teilbaum beim Hauptagenten angemeldet, so muß dieser eingehende SNMP-Pakete in die entsprechenden DPI-Pakete übersetzen und umgekehrt. Dann werden diese Pakete vom Hauptagenten weitergereicht.

Die Implementierung eines Transaktionkonzepts war bei der Entwicklung des SNMP-DPI-Protokolls deshalb von nöten, da mit einem **SNMP-SET** mehrer MIB-Variablen auf einen neuen Wert gesetzt werden können. Sind diese MIB-Variablen aber auf mehrer Subagenten verteilt, so muß der Hauptagent dafür sorgen,daß entweder alle oder keine MIB-Variablen gesetzt werden. Dies ist für die Konsistenzerhaltung der MIB wichtig, da ein Mißerfolg beim Schreiben in einem Subagenten diese nicht gefährden darf.

Kapitel 3

Der neue Hauptagent

3.1 Aufbau und Funktion

Die Weiterentwicklung des Quellcodes des für die Implementierung des Hauptagenten verwendeten Agenten veranlaßte den Lehrstuhl dazu, einen neuen Hauptagenten entwickeln zu lassen. Diese neue Implementierung sollte die Vorteile des neuen UCD-Agenten mit denen eines Hauptagenten vereinen. Der Hauptagent soll Subagenten verwalten können und Information über das System, auf dem er läuft, liefern (siehe Abbildung 3.1). Welche Vorteile bringt der neue Agent?

- Er stellt mehr MIB-Variablen zur Verfügung.
- Er ist modularer aufgebaut, was einer Ergänzung um neue MIB-Variablen vereinfachen soll.

Der neue Hauptagent soll also auf zwei Arten eine Ergänzung um neue MIB-Variablen unterstützen:

- durch den modularen Aufbau des Programmcodes
- und die SNMP-DPI-Schnittstelle.

3.2 Voraussetzungen

Für den neuen Hauptagenten werden folgende Software-Pakete benötigt:

- ucd-snmp-3.3.1.tar.gz [Hardaker]
- hauptagent.tar.gz

Der UCD-Code ist auf folgenden Betriebssystemen[1] lauffähig:

- HP-UX 9.07, 9.05, 9.03, 9.01 auf HPPA 1.1 Systemen
- HP-UX 10.20, 10.10, 10.01 auf HPPA 1.1 Systemen
- Ultrix 4.5, 4.4, 4.3, 4.2 auf DEC MIPS Systemen
- Solaris 2.6, 2.5.1, 2.5, 2.4, 2.3 auf Sun SPARC Systemen
- Solaris 2.5 auf x86 Systemen
- SunOS 4.1.4, 4.1.3, 4.1.3, 4.1.2 auf Sun SPARC Systemen

[1] Diese Angaben stammen aus der Datei ucd-snmp-3.3.1/FAQ (siehe Abschnitt 3.2.1).

- OSF 4.0, 3.2 auf DEC Alpha Systemen
- NetBSD 1.3alpha, 1.2.1, 1.2, 1.1, 1.0 auf allen? Systemen
- FreeBSD 3.0, 2.2.2, 2.2 auf allen? Systemen
- BSDi 2.1 auf allen? Systemen
- Linux 2.1, 2.0, 1.3 auf allen? Systemen
- AIX 4.1.5, 3.2.5 auf allen? Systemen

Der neue Hauptagent wurde vom Praktikanten auf folgenden Systemen erfolgreich kompiliert:

- Linux 2.0.30
- HP-UX 10.20, 9.05, 9.01
- AIX 4.2

Für die unter 3.5 beschriebenen Tests standen jedoch nur Linux-Maschinen zur Verfügung.

3.2.1 Das ucd-snmp-Software-Paket

Das Software-Paket ucd-snmp [Hardaker] wird an der Universität von Californien entwickelt. Es beinhaltet in der aktuellen Version 3.3.1 folgende Teile:

1. Einen erweiterbaren Agenten.
2. Eine SNMP-Bibliothek.
3. Programme zum Setzen und Abfragen von Information von einem SNMP-Agenten.
4. Programme zum Generieren und Empfangen von SNMP-Traps.
5. Eine Version des UNIX-Kommandos `netstat`, daß SNMP verwendet.

Das Software-Paket kann mit folgenden Kommandos entpackt werden:

- `tar xzf ucd-snmp-3.3.1.tar.gz` (unter Linux)
- `gunzip ucd-snmp-3.3.1.tar.gz; tar xf ucd-snmp-3.3.1.tar` (unter HP-UX)

Das entpackte Software-Paket liegt dann im Unterverzeichnis mit dem Namen ucd-snmp-3.3.1/ des aktuellen Verzeichnisses. Dort sind auch weitere wichtige Informationsquellen zu finden, welche folgende Dateien wären: README, FAQ, INSTALL und PORTING. Weitere wichtige Unterverzeichnisse von ucd-snmp-3.3.1/ sind:

- agent/
  Dort ist der Quelltext des UCD-Agenten zu finden, nach der Kompilierung auch der ausführbare Agent namens snmpd.
- apps/
  Dies ist das Verzeichnis mit allen Programmen von 3. und 4..
- snmplib/
  Hier liegt eine SNMP-Bibliothek.

- etc/
  Um mit sicheren SNMP Paketen zu kommunizieren, müssen für das System Sicherheitsparameter gesetzt werden. Hier liegen Dateien für eine Beispiel-Konfiguration.
- mibs/
  Hier liegen Textdateien in ASN.1-Syntax zur Beschreibung der Internet-MIBs. (in Abschnitt 3.2.2 wird ihre Bedeutung erklärt).
- ov/
  Dieses Verzeichnis enthält Dateien, die die Verwendung von HP-OpenView unterstützen.
- win32/
  Dateien für die Erzeugung von Windows95- und WindowsNT-Versionen der im Verzeichnis apps/ liegenden Programme.

3.2.2 Das hauptagent-Software-Paket

Für den Bau des Hauptagenten aus dem UCD-Agenten wird zusätzlich das Software-Paket hauptagent.tar.gz benötigt. Es besteht aus folgenden Dateien:

- hauptagentpatch
  Patch-Datei, mit der alle nötigen Änderungen für die Erzeugung des Hauptagenten an ucd-snmp-3.3.1-Dateien durchführen werden.
- makehauptagent
  Ein Shell-Script als Alternative zur Patch-Datei.
- move
  Dieses Shell-Script verschiebt die nachfolgenden Dateien in die richtigen Verzeichnisse.
- dpi.c
  Beinhaltet die Implementierung der SNMP-DPI-Schnittstelle im Hauptagenten [Hain 94].
- dpi.h
  Die zur Datei dpi.c gehörige Header-Datei [Hain 94].
- mibgroup_dpi.h
  Diese Datei enthält die Ergänzung des Hauptagenten um die DPI20-MIB.
- mibgroup_dpi.c
  Hier sollte die Implementierung der DPI20-MIB-Ergänzung stehen, jedoch sind diese schon in der Datei dpi.c Implementiert. Daher kann diese Datei leer bleiben. Sie muß aber vorhanden sein.
- snmp_agent_in.c
  Komplexere Änderungen werden vom Script makehauptagent nicht durchgeführt, Funktionen in denen sie nötig sind werden einfach ersetzt. Diese und die nächste Datei enthalten diese neueren Funktionen. (Nähere Information kann unter 4.2 und 4.3 gefunden werden)
- snmp_vars_in.c
- snmp_dpi.h
  Header-Datei für die SNMP-DPI-API[2] [Hain 94].

[2] Application Programming Interface

- snmp_lDPI.h
  Kleine Ergänzung der vorigen Header-Datei [Hain 94].
- snmp_mDPI.c
  SNMP-DPI-API [Hain 94]
- DPI20-MIB.txt
  DPI20-MIB in ASN.1-Syntax (siehe Anhang A). Wird nicht unbedingt benötigt. Jedoch kann sie und die MIB-Dateien in mibs/ von snmpwalk u. a. verwendet werden, um die numerischen Identifikatoren der DPI20-MIB-Variablen durch aussagekräftigere Bezeichner (iso.org.dod.internet.private.enterprise.ibm.ibmDPI-.dpi20MIB.dpiPort.dpiPortForTCP statt 1.3.6.1.4.1.2.2.1.1.1.0) zu ersetzen. Dies gilt auch bei der Verwendung der Option `-d` bei ./snmpd. Erreicht wird dies durch Setzen der Umgebungsvariable `MIBS` auf den Wert `ALL`.

3.3 Erzeugung des Hauptagenten

Zur Erzeugung des Hauptagent wird folgendermaße vorgegangen:

1. Kopieren des ucd-snmp-Softwarepaket dorthin, wo das Verzeichnis ucd-snmp-3.3.1/ erzeugt werden soll.
2. Entpacken des Softwarepaket ucd-snmp-3.3.1.tar.gz.
3. Kopieren der Datei hauptagent.tar.gz in das Verzeichnis mit dem Agenten (normalerweise ucd-snmp-3.3.1/).
4. Entpacken von hauptagent.tar.gz.
5. Den UCD-Code durch eine der beiden Möglichkeiten ändern:
   (a) Aufrufen von `makehauptagent` (Fehler werden nicht angezeigt).
   (b) Folgende Befehlsfolge ausführen: `cp hauptagentpatch ../; cd ..; patch <hauptagentpatch` (Nicht durchgeführte Änderungen legt das Programm in Dateien mit der Endung .rej ab).
6. Aufrufen von `move`.
7. Den Agenten konfigurieren mit `./configure --with-mib-modules="dpi"`
8. Ausführen von `make`.

Bei 7. muß gegebenenfalls der Installationspfad noch mit angegeben werden, falls er vom Standardpfad (/usr/local/) abweicht. Dies geschieht mit:
`--prefix=<PATH>` (z.B. `--prefix="/users/home/friedrim/"`)

Ähnliches gilt für das Standardverzeichnis für die MIB-Dateien. Dieses wird mit `--with-mibdirs="<dir1>:<dir2>"` eingestellt.

Für eine Installation im Verzeichnis /users/home/friedrim/ würde sich zum Beispiel folgender Befehl ergeben:

```
./configure --prefix="/users/home/friedrim/"
--with-mibdirs="/users/home/friedrim/ucd-snmp-3.3.1/mibs"
--with-mib-modules="dpi"
```

Nähere Information kann der Datei ucd-snmp-3.3.1/INSTALL entnommen werden.

3.4 Installation des Hauptagenten

Der neue Hauptagent befindet sich nun im Unterverzeichnis agent und heißt snmpd.

Bevor er ausgeführt wird, sollte zuerst die Datei trap.conf im Verzeichnis /etc/ angelegt werden. Diese Datei hat folgenden Aufbau:
<Hostname> <IP-Adresse> <Community-Name>.
z.B.

```
hpheger2      129.187.214.22    public
hpheger3      129.187.214.23    public
hpheger4      129.187.214.24    public
hpheger7      129.187.214.27    public
ibmhegering1  129.187.214.19    public
```

Diese Datei enthält alle Rechner, an die der Hauptagent Traps an den Port 162 schicken soll. Die Bedeutung des Community-Name wird in Abschnitt 3.5.1 erklärt.

Der Agent wird dann durch `./snmpd` gestartet. Mit `./snmpd -h` können die Optionen, die der Hauptagent entgegen nimmt, abgefragt werden. Auf ein paar Optionen soll jedoch kurz eingegangen werden:

- `-d` dies veranlaßt den Hauptagenten alle gesendeten und empfangenen Pakete und die durchgeführten Aktionen auf dem Bildschirm auszugeben.
- `-L` mit dieser Option werden alle Nachrichten, die normalerweise dem Logfile angehängt werden, auf die Standardausgabe geschrieben.
- `-C` läßt den Hauptagenten die Konfigurationsdateien ignorieren.
- `-p <NUM>` der Hauptagent verwendet statt Port 161 den Port `<NUM>` für die Entgegennahme von SNMP-Paketen. Mit dieser Option kann man den Agenten ohne root-Rechte starten. Ports unter 1000 verlangen root-Rechte.

3.5 Test des Hauptagenten

3.5.1 Beschreibung der Testtools

Der Hauptagent kann mittels der im Verzeichnis apps/ befindlichen Programme getestet werden. Hier einige Beispiele:

- snmpget
  Mit diesem Programm werden SNMP-Variablen abgefragt.
- snmpwalk
  Wie snmpget mit dem Unterschied, daß ein kompletter Unterbaum der Internet-MIB abgefragt wird.
  Z.B. `./snmpwalk -v 1 localhost public .1.3.6`
- snmpset
  Hiermit lassen sich SNMP-Variablen setzen.
- snmptrapd
  Damit können SNMP-Traps empfangen werden. Dieses Programm muß von root gestartet werden und auf einer der Maschinen aus der Datei trap.conf laufen.
  Z.B. `./snmptrapd -P` Die Option `-P` sorgt für die Ausgabe am Bildschirm.

Da der Hauptagent (snmpd) auf einer anderen Maschine laufen kann als das verwendete Testprogramm, muß diesem Testprogramm der Rechner mitgeteilt werden auf dem der Hauptagent läuft. `localhost` gibt an, daß beide Programme auf dem selben Rechner ablaufen.

Bei `public` handelt es sich um den Community-Namen für die Transaktion. Er dient als eine Art Paßwort für die durchgeführte Operation. Stimmt dieser mit dem entsprechenden Community-Namen beim Agenten (Subagenten) überein, so wird die Erlaubnis für die Operation gewährt.

3.5.2 Test mit syslogtrapd

Hier soll anhand eines Beispieles gezeigt werden, wie die SNMP-DPI-Schnittstelle des Hauptagent getestet werden kann. Für diesen Zweck wird der Subagent syslogtrapd verwendet, der von Christian Coehn an der Technischen Universität München entwickelt wurde. Ziel dieses Programmes ist es den Syslog-Daemon, der auf allen UNIX-Systemen läuft, mittels SNMP zu managen [Coeh 97].

Der Syslog-Daemon bietet Anwendungs- und Systemprogrammen eine standartisierte Schnittstelle, um Nachrichten, Warnungen und Fehler zu protokollieren. Programme mit Meldungen schicken in der Regel diese an den Socket[3] /dev/log, an dem sie der Syslog-Daemon in Empfang nimmt (siehe Abbildung 3.2).

Damit nun der Subagent ankommende Meldungen registriert, wird er anstelle des Syslog-Daemons an diesen Socket gehängt. Der Subagent kann nun auf diese Meldungen reagieren und anschließend diese an den Syslog-Daemon weiterreichen. Für die Kommunikation zwischen Subagent und Syslog-Daemon wird ein Socket eingerichtet (hier /dev/log.syslog; siehe Abbildung 3.3).

Der Test soll nun folgendermaßen aussehen: der Subagent wird so konfiguriert, daß er bei einem Einlog-Versuch einen Trap an den Hauptagenten schickt und dieser ihn weiterreicht an das Programm snmptrapd, von dem er dann angezeigt wird.

1. Falls der Syslog-Daemon schon läuft, muß er beendet werden.
2. Falls vorhanden, muß die Datei /dev/log gelöscht werden.
3. Falls die Datei /etc/trap.conf nicht vorhanden ist muß sie erzeugt werden.
4. Starten des Hauptagenten mit `snmpd -d -L`.
5. Starten des Subagenten mit `perl syslogtrapd -d`.
6. Starten des Syslog-Daemon mit `syslogd -p /dev/log.syslog`.
7. Den Subagenten konfigurieren mit der Befehlsfolge
```
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.1.1.2.0 i 5
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.1.1.1.0 i 1
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.1.1.3.0 s "auth"
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.1.1.4.0 s ".*"
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.1.1.5.0 s ".*"
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.1.1.2.0 i 1
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.3.1.4.0 i 5
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.3.1.1.0 i 1
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.3.1.2.0 i 1
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.3.1.3.0 i 1
snmpset -v 1 localhost public .1.3.6.1.3.100.8.1.1.3.1.4.0 i 1
```
8. Durch `snmptrapd -P` werden alle ankommenden Traps angezeigt.
9. Versuchen sich unter root einzuloggen.

Für die Programme unter 4.,5. und 8. soll jeweils eine eigenen Shell gestartet werden, damit deren Ausgabe mitverfolgt werden kann (siehe Abbildung 3.4 und 3.5).

[3] manche UNIX-Systeme verwenden stattdessen Pipes z.B. HP-UX

Mit folgendem Befehl können alle MIB-Variablen, die `syslogtrapd` verwaltet, abgefragt werden:

```
snmpwalk -v 1 localhost public .1.3.6.1.3.100.8.1
```

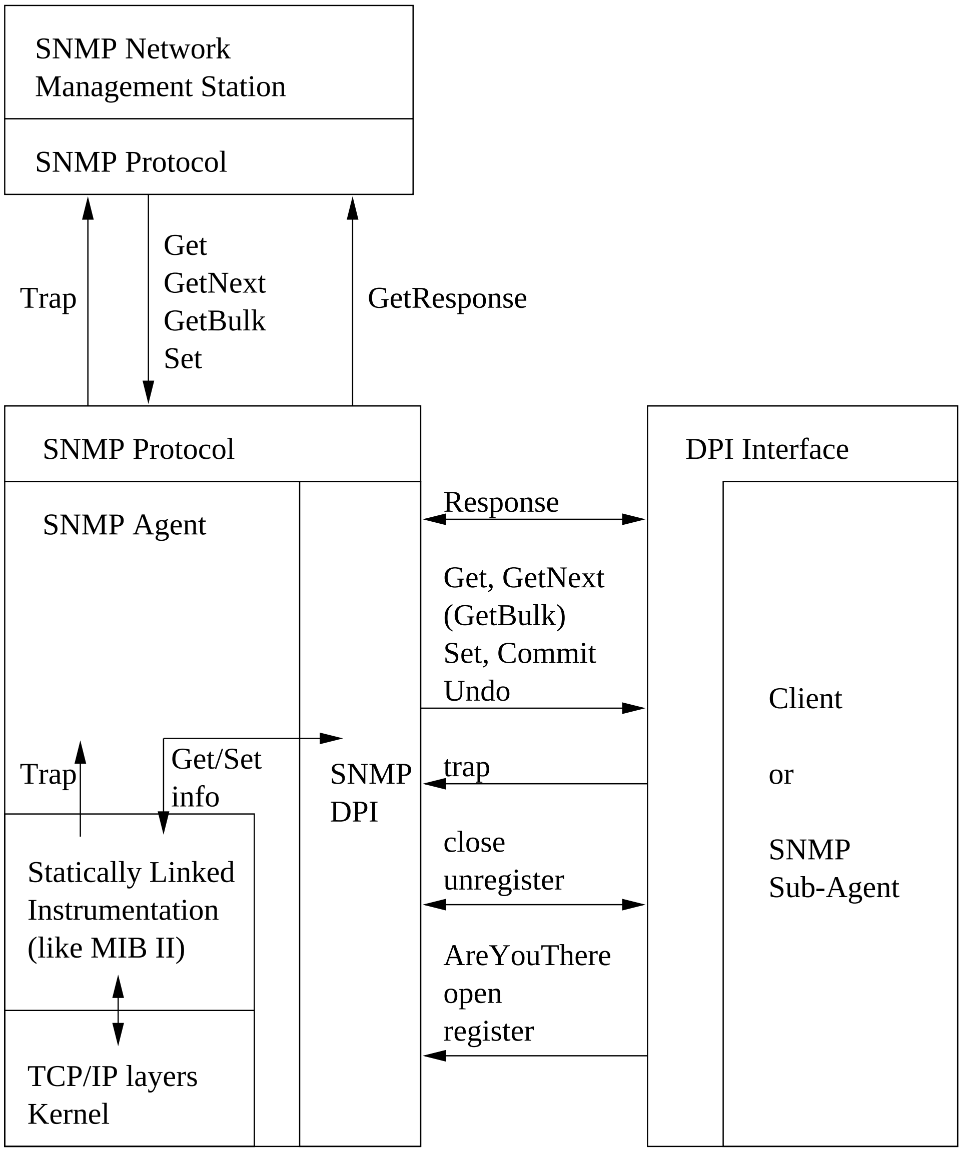

Abbildung 3.1: SNMP-DPI

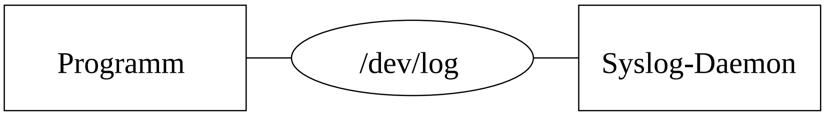

Abbildung 3.2:

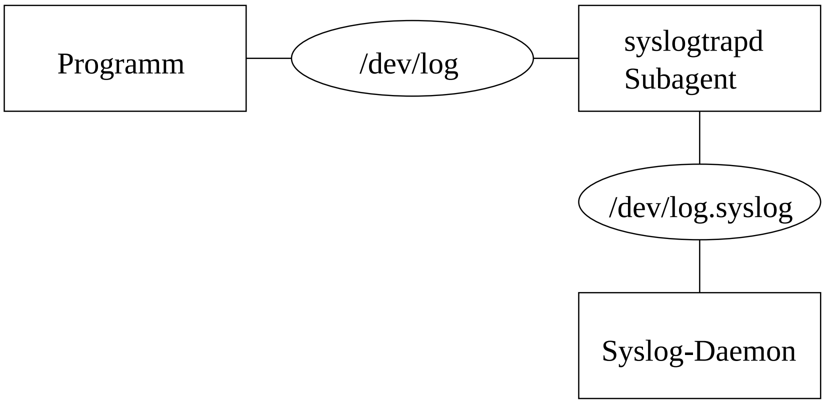

Abbildung 3.3:

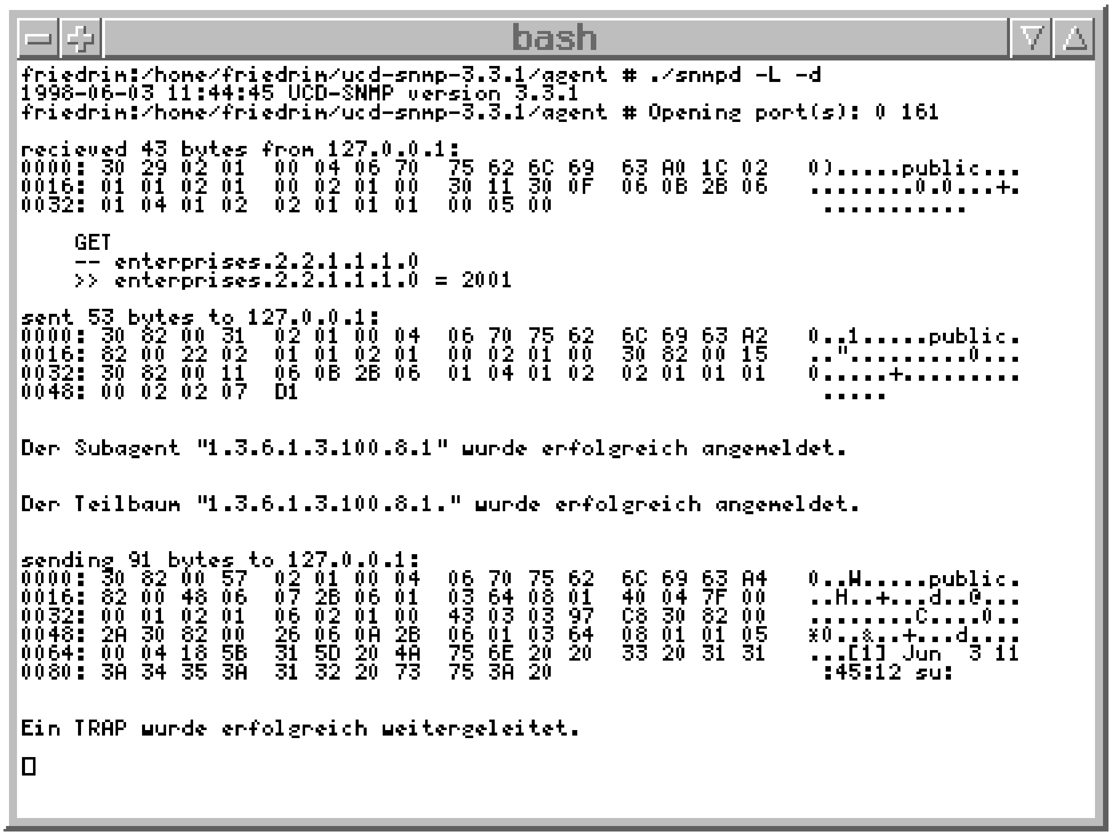

Abbildung 3.4:

Abbildung 3.5:

Kapitel 4

Details der Implementierung

In den folgenden Abschnitten werden die Änderungen vorgestellt, die mit der Patch-Datei hauptagentpatch oder dem Shell-Script makehauptagent an folgenden Dateien durchgeführt werden:

1. agent/snmpd.c
2. agent/snmp_agent.c
3. agent/snmp_vars.c
4. agent/Makefile
5. snmplib/snmp_api.h
6. snmplib/system.h
7. snmplib/snmp_api.c

Dieses Kapitel kann auch als eine Anleitung für die manuelle Erzeugung des Hauptagenten aufgefaßt werden.

Das Konzept, daß hinter den Änderungen am UCD-Agenten steckt, beruht auf einer Einteilung der Kommunikationen zwischen Manager, Hauptagent und Subagenten. Man unterscheidet hierbei eine Kommunikation, die vom Manager und eine die vom Subagenten eingeleitet wird. Der Subagent wird nur dann von sich aus aktiv, wenn er einen MIB-Teilbaum registrieren, deregistreieren oder einen Trap verschicken will. Der UCD-Agent muß nun um diesen Kommunikationsweg erweitert werden, dies ist in Abschnitt 4.1 zu finden.

Das Beantworten der Anfragen vom Manager ist hingegen keine neue Aufgabe des Hauptagenten, jedoch muß er so verändert werden, daß er auch Zugriff auf die MIB-Variablen der Subagenten hat. Die entsprechenden Änderungen werden in Abschnitt 4.2 und 4.3 erklärt.

4.1 Änderung in der Datei snmpd.c

Die Datei snmpd.c enthält das Hauptprogramm des UCD-Agenten. Da hier bisher nur die Manager-Agenten-Kommunikation implementiert ist, muß sie um die Subagent-Agent-Kommunikation erweitert werden. Die Funktion *receive* ist beim Agenten für den Empfang von eingehenden Paketen zuständig. Durch die neuen Funktionen *reading_data_from_tcpsockets* und *reading_data_from_udpsockets* wird sie nun um diesen neuen Kommunikationsweg ergänzt.

Funktion *receive*

```
...
            default:
                printf("select returned %d\n", count);
                return -1;
        }

        reading_data_from_tcpsockets();
        reading_data_from_udpsockets();

        gettimeofday(nvp, (struct timezone *) NULL);
        if (nvp->tv_sec > svp->tv_sec
            || (nvp->tv_sec == svp->tv_sec && nvp->tv_usec > svp->tv_usec)){
#ifdef USING_ALARM_MODULE
            alarmTimer(nvp);
...
```

Für den Betrieb dieser Funktionen muß noch eine kleine Vorarbeit in der Funktion *main* geleistet werden.

Funktion *main*

```
...
    starttime.tv_usec += 1000000L;

    /* send coldstart trap via snmptrap(1) if possible */
    send_easy_trap (0);

    if (komm_init() < 0) {
       fprintf(stderr, "Fehler beim DPI-Socket einrichten.\n");
       exit(1);
    }

    if (!read_trap_database("/etc/trap.conf")) {
       if (snmp_dump_packet)
          fprintf(stderr, "Fehler beim Einlesen der Zielrechner fuer SNMPv1 TRAPs.\n");
    }

    interne_teilbaeume_registrieren();


    memset(addrCache, 0, sizeof(addrCache));
    receive(sdlist, sdlen);
...
```

Die Deklarationen dieser neuen Funktionen sind in der Datei dpi.h zu finden. Diese Deklarationen werden folgendermaßen in die Datei snmpd.c eingefügt: der letzten include-Anweisung wird die Anweisung `#include "dpi.h"` angehängt.

Dieser Schritt muß auch für die Dateien in Abschnitt 4.2 und 4.3 durchgeführt werden.

Eine weitere Änderung betrifft die Anweisung

```
int snmp_dump_packet;
```

die zu

```
extern int snmp_dump_packet;
```

wird. Der Grund hierfür kann in Abschnitt 4.7 gefunden werden.

4.2 Änderung in der Datei snmp_agent.c

Die Bearbeitung von eingehenden Anfragen vom Manager übernehmen beim Hauptagenten zwei Funktionen, die in der Datei snmp_agent.c spezifiziert werden. Sie erstellen Antwortpakete zu diesen Anfragen. Für **GETBULK**-Anfragen ist die Funktion *bulk_var_op_list* zuständig, alle anderen Anfragen werden von *parse_var_op_list* erledigt. Da nun MIB-Variablen sowohl im Hauptagenten als auch im Subagenten gehalten werden, müssen diese Funktionen so verändert werden, daß sie auch auf MIB-Variablen der Subagenten zugreifen können. Neuen Versionen dieser Funktionen sind in der Datei snmp_agent_in.c zu finden und werden in die Datei snmp_agent.c eingefügt.

```
...
struct repeater {
    oid name[MAX_NAME_LEN];
    int length;
} repeaterList[10];

#include "snmp_agent_in.c"
...
```

Die Beseitigung der alten Funktionen erfolgt durch einfache Umbennung.

```
static int
parse_var_op_list(data, length, out_data, out_length, index, pi, action)
```

wird zu

```
static int
old_parse_var_op_list(data, length, out_data, out_length, index, pi, action)
```

und

```
static int
bulk_var_op_list(data, length, out_data, out_length, non_repeaters,
                 max_repetitions, index, pi)
```

wird zu

```
static int
old_bulk_var_op_list(data, length, out_data, out_length, non_repeaters,
                 max_repetitions, index, pi)
```

4.3 Änderung in der Datei snmp_vars.c

Die Datei snmp_vars.c enthält Funktionen für die Verwaltung der MIB-Variablen des UCD-Agenten, besonders die Funktion *getStatPtr* für den Zugriff auf die MIB-Variablen. Sie ist ein wichtiges Werkzeug der Funktionen aus dem vorigen Abschnitt. Diese Funktion wird von der Funktion *mnm_getStatPtr* ersetzt, welche zusätzlich noch auf die MIB-Variablen der Subagenten zugreifen kann. Sie und die Funktion *getStatPtr*[1] stehen in der Datei snmp_vars_in.c. *getStatPtr* wird verwendet, wenn intern im Hauptagenten auf MIB-Variablen zugegriffen werden muß. Da intern nur **GET**-Anfragen durchgeführt werden dürfen, stellt sie eine Einschränkung von *mnm_getStatPtr* dar. An folgender Stelle wird nun die Datei mit diesen Funktionen eingefügt:

```
...
#ifdef USING_VIEW_VARS_MODULE
extern int in_view __P((oid *, int, int));
#endif

int subtree_old_size() {
  return (sizeof(subtrees_old)/ sizeof(struct subtree));
}
...
```

Dies ergibt sich zu folgendem Progammcode:

[1] Es handelt sich hier nicht um *getStatPtr* des UCD-Agenten

```
...
#ifdef USING_VIEW_VARS_MODULE
extern int in_view __P((oid *, int, int));
#endif

#include "snmp_vars_in.c"

int subtree_old_size() {
  return (sizeof(subtrees_old)/ sizeof(struct subtree));
}
...
```

Analog zum vorigen Abschnitt wird die alte Funktion beseitigt.

```
u_char  *
getStatPtr(name, namelen, type, len, acl, exact, write_method, pi,
           noSuchObject)
```

wird zu

```
u_char  *
old_getStatPtr(name, namelen, type, len, acl, exact, write_method, pi,
           noSuchObject)
```

4.4 Änderung in der Datei Makefile.in

Für den Zusammenbau des Hauptagenten wird das Makefile.in folgendermaßen erweitert:

```
...
getmibstat.o: mibgroup/kernel_sunos5.c
        $(CC) $(CFLAGS) -o $@ -D_GETMIBSTAT_TEST -DDODEBUG -c $?

AGENTOBJS=snmpd.o kernel.o snmp_agent.o snmp_vars.o view_parse.o read_config.o

snmpd:  ${MIBOBJS} ${AGENTOBJS} ../snmplib/libsnmp.a
        ${CC} -o $@ ${AGENTOBJS} ${MIBOBJS} ${LDFLAGS} ${LIBS}
...
```

zu

```
...
getmibstat.o: mibgroup/kernel_sunos5.c
        $(CC) $(CFLAGS) -o $@ -D_GETMIBSTAT_TEST -DDODEBUG -c $?

snmp_mDPI.o:
        $(CC) $(CFLAGS) -o $@ -c snmp_mDPI.c

dpi.o:
        $(CC) $(CFLAGS) -o $@ -c dpi.c

AGENTOBJS=snmpd.o kernel.o snmp_agent.o snmp_vars.o view_parse.o dpi.o snmp_mDPI
.o read_config.o

snmpd:  ${MIBOBJS} ${AGENTOBJS} ../snmplib/libsnmp.a
        ${CC} -o $@ ${AGENTOBJS} ${MIBOBJS} ${LDFLAGS} ${LIBS}

...
```

Nachfolgende Änderungen waren nötig, um die unterschiedlichen Programmteile — UCD-Agent und SNMP-DPI-Schnittstelle — zu verbinden.

4.5 Änderung in der Datei snmp_api.h

```
...
/*
```

```
extern int snmp_dump_packet;
extern int quick_print;
*/
...
```

wird zu

```
...
extern int snmp_dump_packet;
/*
extern int quick_print;
*/
...
```

4.6 Änderung in der Datei system.h

```
char *strdup __P((char *));
```

wird zu

```
char *strdup __P((__const char *__s));
```

4.7 Änderung in der Datei snmp_api.c

```
static int snmp_dump_packet = 0;
```

wird zu

```
int snmp_dump_packet;
```

Diese Änderung zusammen mit der im Abschnitt 4.1 beschriebenen Änderung dieser Variable war notwendig, da sie sonst in beiden Dateien nicht identisch ist, und damit ein Teil der Dump-Ausgabe verloren geht.

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

Bei dem Zusammenbau des Hauptagenten aus Agent und bereits bestehender Dateien konnte man den Aufwand sich in den Quellcode einzuarbeiten sehr gut abschätzen. Da die Einarbeitung in den Quellcode des Agenten einen großen Teil dieser Arbeit ausmachte und dies sehr zeitaufwendig war, kann ich die Vorteile die das SNMP-DPI-Protokoll durch den Wegfall dieser Tätigkeit mit sich bringt gut nachvollziehen.

Im Laufe dieser Arbeit wurde dem Praktikanten das im Internet-Management verwendete Protokoll SNMP und die damit bestehenden Möglichkeiten und Grenzen für das Management von Rechnernetzen an praktischen Beispielen aufgezeigt. Dies gilt natürlich auch für das SNMP-DPI-Protokoll.

Interessant war es auch einmal einen Quellcode eines Programms für unterschiedlich UNIX-Varianten kennenzulernen. Hier konnte man abschätzen, wie wohl ein POSIX-UNIX die Softwareentwicklung erleichtern würde.

Es hinterließ nicht nur das Arbeiten mit den verschiedenen UNIX-Betriebssystemen, sondern auch der Umgang mit verschiedenen C-Compilern einen bleibenden Eindruck. Dies betrifft hauptsächlich die Strenge im Umgang mit verschiedenen C-Standards. Es stellte sich heraus, daß es für manchen C-Compiler unmöglich war Dateien mit ANSI-C Code mit nicht-ANSI-C Code zu verbinden. Dagegen war dies für einen verwendeten Compiler kein Problem.

5.2 Ausblick

Heute wird das SNMP-Protokoll hauptsächlich zum Managen von Rechnernetzen genutzt. Mit dem SNMP-DPI-Protokoll ist den Entwicklern von Softwaresystemen jedoch die Möglichkeit gegeben ihre eigenen MIB-Variablen einzuführen, und damit ihre Softwaresysteme über SNMP konfigurierbar zu machen. Dies kann der Verbreitung des SNMP-Systemmanagement [Haha 94] nur dienen.

Das SNMP-DPI-Protokoll wurde als ein Experiment für erweiterbare Agenten von IBM entwickelt. Aus den Erfahrungen, die IBM aus diesem Experiment erlangte, entstand kürzlich das AgentX-Protokoll [Daniele 98]. Dieses besitzt neben einer größeren Funktionaltität auch noch eine AgentX-MIB [Greene 98], mit der man das AgentX-Protokoll überwachen und konfigurieren kann. Ein gravierender Unterschied zu SNMP-DPI liegt darin, daß die Prozeßkommunikation von Agent und Subagent nicht an TCP/IP gebunden ist.

Es ist zu hoffen, daß in Zukunft bei der Entwicklung neuer Software das SNMP-Management berücksichtigt wird. Natürlich kann man oft auch nachträglich Programme SNMP-managebar zu machen, wie das Beispiel in 3.5.2 zeigt.

Anhang A

DPI20-MIB.txt

```
DPI20-MIB DEFINITIONS ::= BEGIN

-- Objects in this MIB are implemented in the local SNMP agent.

   IMPORTS
           MODULE-IDENTITY, OBJECT-TYPE, snmpModules, enterprises
                   FROM SNMPv2-SMI

   ibm      OBJECT IDENTIFIER ::= { enterprises 2 }
   ibmDPI   OBJECT IDENTIFIER ::= { ibm 2 }
   dpi20MIB OBJECT IDENTIFIER ::= { ibmDPI 1 }

-- dpi20MIB MODULE-IDENTITY
--   LAST-UPDATED "9401210000Z"
--   ORGANIZATION "IBM Research - T.J. Watson Research Center"
--   CONTACT-INFO "           Bert Wijnen
--                 Postal:    IBM International Operations
--                            Watsonweg 2
--                            1423 ND Uithoorn
--                            The Netherlands
--                 Tel:       +31 2975 53316
--                 Fax:       +31 2975 62468
--                 E-mail:    wijnen@vnet.ibm.com"
--   DESCRIPTION "MIB module describing DPI objects."
--   ::= { snmpModules x }

   dpiPort  OBJECT IDENTIFIER ::= { dpi20MIB 1 }

   dpiPortForTCP   OBJECT-TYPE
           SYNTAX  INTEGER (0..65535)
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION "The TCP port number on which the agent
                        listens for DPI connections. A zero value
                        means the agent has no DPI TCP port."
           ::= { dpiPort 1 }

   dpiPortForUDP   OBJECT-TYPE
           SYNTAX  INTEGER (0..65535)
```

```
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION "The UDP port number on which the agent
                        listens for DPI packets. A zero value
                        means the agent has no DPI UDP port."
           ::= { dpiPort 2 }
END
```

Literaturverzeichnis

[Carpenter 91] G. Carpenter und B. Wijnen, „RFC 1228: SNMP-DPI: Simple Network Management Protocol Distributed Program Interface“, Technischer Bericht, IETF, Mai 1991, http://rfc.fh-koeln.de/rfc/html/rfc1228.html oder ftp://ftp.leo.org/pub/comp/doc/standards/rfc/rfc-1200-1299/rfc1228.

[Case 90] J. D. Case, M. Fedor, M. L. Schoffstall und C. Davin, „RFC 1157: Simple Network Management Protocol (SNMP)“, Technischer Bericht, IETF, Mai 1990, http://rfc.fh-koeln.de/rfc/html/rfc1157.html oder ftp://ftp.leo.org/pub/comp/doc/standards/rfc/rfc-1100-1199/rfc1157.

[Case 93] J. Case, K. McCloghrie, M. Rose und S. Waldbusser, „RFC 1441: Introduction to version 2 of the Internet-standard Network Management Framework“, Technischer Bericht, IETF, April 1993, http://rfc.fh-koeln.de/rfc/html/rfc1441.html oder ftp://ftp.leo.org/pub/comp/doc/standards/rfc/rfc-1400-1499/rfc1441.

[Coeh 97] C. Coehn, „Entwicklung einer Managementschnittstelle für den Syslog-Dämon“, Fortgeschrittenenpraktikum, Ludwig-Maximilians-Universität München, Oktober 1997.

[Daniele 98] M. Daniele, B. Wijnen und D. Francisco, „RFC 2257: Agent Extensibility (AgentX) Protocol Version 1“, Technischer Bericht, IETF, Januar 1998, http://rfc.fh-koeln.de/rfc/html/rfc2257.html oder ftp://ftp.leo.org/pub/comp/doc/standards/rfc/rfc-2200-2299/rfc2257.

[Greene 98] Maria Greene und S. Gudur, „Definitions of Managed Objects for Extensible SNMP Agents“, Internet Draft, IETF, 04 1998, gopher://ds0.internic.net:70/00/internet-drafts/draft-ietf-agentx-mib-02.txt oder http://info.internet.isi.edu:/0/in-drafts/files/draft-ietf-agentx-mib-02.txt.

[HA 93] H.-G. Hegering und S. Abeck, *Integriertes Netz- und Systemmanagement*, Addison-Wesley, Maerz 1993.

[Haha 94] N. Haubelt und R. Hauck, „Implementierung einer MIB für Systemmanagementaufgaben“, Fortgeschrittenenpraktikum, Technische Universität München, Dezember 1994.

[Hain 94] E. Hainzinger, „Erweiterung eines SNMPv2-Agenten um eine Schnittstelle zur Interprozeßkommunikation“, Fortgeschrittenenpraktikum, Technische Universität München, Dezember 1994.

[Halsall 95] Fred Halsall, *Data Communications, Computer Networks and Open Systems*, Addison-Wesley, 4. Auflage, 1995.

[Hardaker] Wes Hardaker, *UCD SNMP package*, ftp:/ftp.ece.ucdavis.edu/pub/snmp/ucd-snmp.tar.gz oder ftp://sunsite.cnlab-switch.ch/mirror/ucd-snmp/ucd-snmp.tar.gz oder ftp://ftp.win.or.jp/network/snmp/ucd-snmp/ucd-snmp.tar.gz.

[Kofler 95] Michael Kofler, *Linux Installation, Konfiguration, Anwendung*, Addison-Wesley, 2. Auflage, 1995.

[Kopka 96] Helmut Kopka, *LaTeX-Einführung*, Addison-Wesley, 2. Auflage, 1996.

[Wijnen 94] B. Wijnen, G. Carpenter, K. Curran, A. Sehgal und G. Waters, „RFC 1592: Simple Network Management Protocol Distributed Protocol Interface Version 2.0“, Technischer Bericht, IETF, Maerz 1994, http://rfc.fh-koeln.de/rfc/html/rfc1592.html oder ftp://ftp.leo.org/pub/comp/doc/standards/rfc/rfc-1500-1599/rfc1592.