Praktikum SNMP – Versuch

FH München
12. Februar 2003

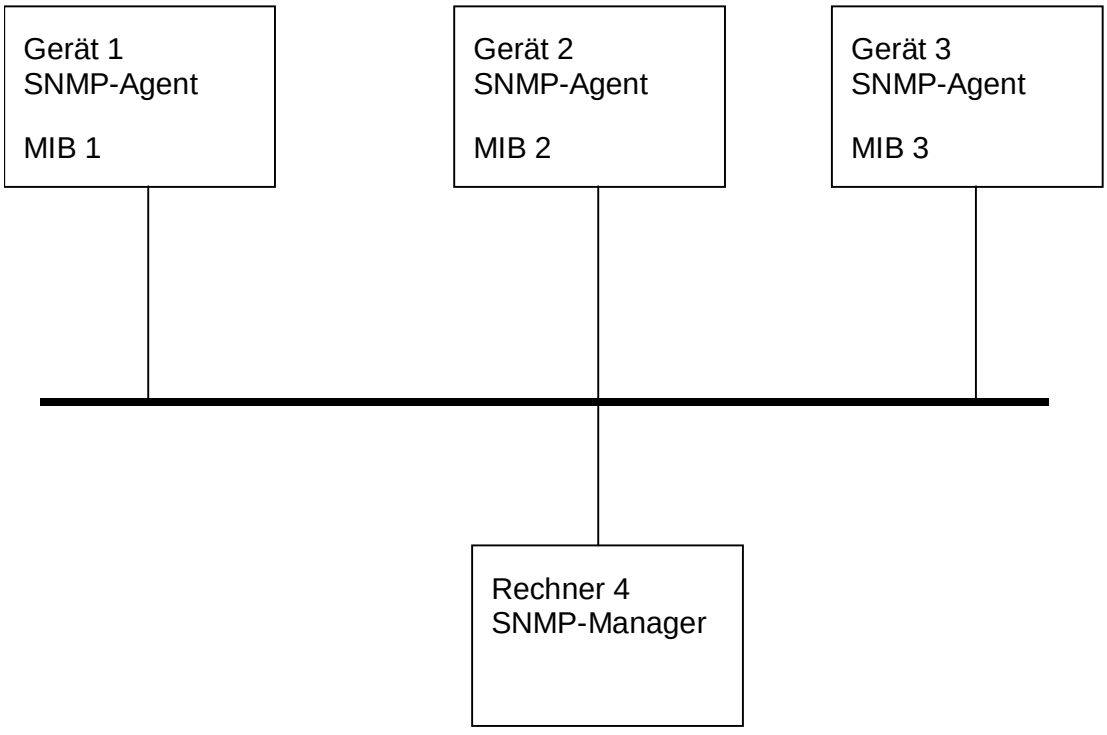
Dr.sc.nat.
Michael J.M. Wagner

1 Einleitung

Rechnernetze sind vom privaten Bereich über die Forschung zum gewerblichen Bereich allgegenwärtig. Ein entscheidender Kostenfaktor für den Betrieb von Rechnernetzen ist deren Wartung.

Ein Protokoll zur Fernwartung von Rechnernetzen auf TCP/IP-Basis ist das *Simple Network Management Protocol* (SNMP).

In der Vorlesung wurden bereits die grundlegenden Konzepte, sowie das Berechtigungskonzept besprochen. Hier noch einmal die Struktur:



Auf den zu überwachenden Geräten – hier ist alles mögliche denkbar, das einen TCP/IPAnschluss hat und SNMP unterstützt – sind Agenten installiert, die das eigene Gerät und dessen Netzwerkumgebung ausforschen und die Daten in einer Datenbasis, der *Management Information Base* (MIB) ablegen. Von einem zentralen Rechner, dem SNMP-Manager, können nun diese Datenbasen ausgelesen, aber auch verändert werden. Das SNMP sieht dafür die Operationen *get, get-next* und *set*

vor. Dazu werden entsprechende *Protocol Data Units* (PDU) an die Agenten geschickt, die diese mit PDU-RESPONSE beantworten.

Darüber hinaus können Agenten mit *SNMP traps* akut auftretende Probleme an den Manager melden. Hierzu ist allerdings zu bemerken, dass diese von vielen Agenten – so wie beispielsweise von dem in diesem Praktikum verwendeten UCD-Agenten – nicht unterstützt werden.

2 Die Management Information Base (MIB)

Welche Daten werden nun in einer MIB gehalten ?

Das hängt natürlich von der Art des zu überwachenden Geräts ab (ein Kühlschrank wird andere Daten ablegen, als ein Drucker oder Computer), sowie vom eingesetzten Agenten.

Die Information der MIB ist baumartig organisiert. Damit verschiedene Manager und Agenten verschiedener Hersteller eine gemeinsame Sprache über die auszutauschende Information finden, werden die Knoten des Baumes bei der ISO (für das SNMP) bzw. der CCITT (im Telekommunikationsbereich) registriert. Der Baum startet mit dem Wurzelknoten „.“ (analog dem Wurzelverzeichnis „/“ bei UNIX). Ein Knoten in diesem Baum wird durch eine Folge natürlicher Zahlen (z.B. „.1.3.6.1.2.1“) identifiziert. Darüber hinaus sind folgende Schreibweisen üblich:

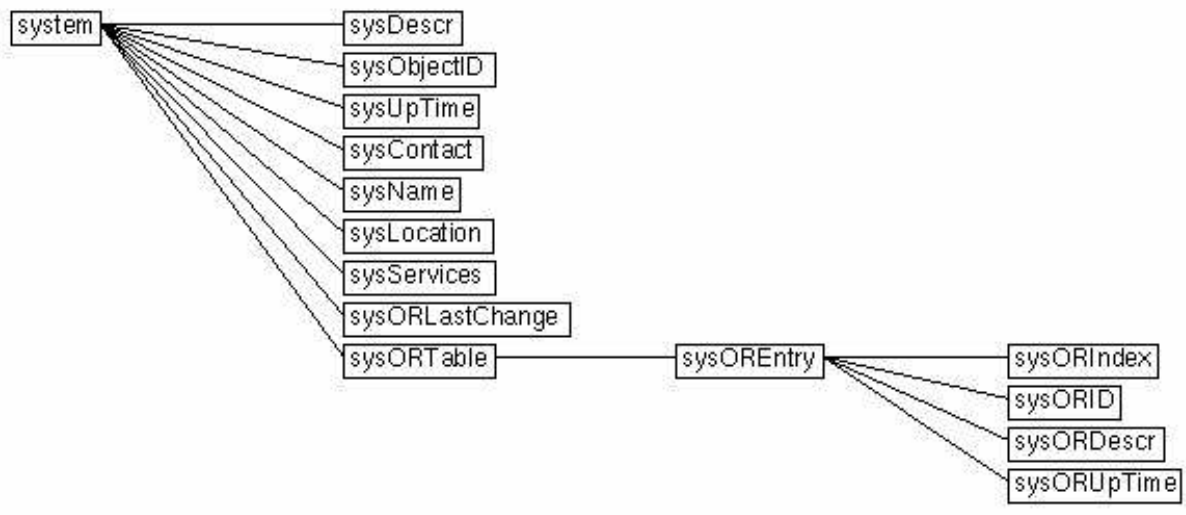
```
.iso.org.dod.internet.mgmt.mib-2

{ iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) }

{ 1 3 6 1 2 1 }
```

Die Bedeutung einzelner Knoten wird in ASN.1 (Abstract Syntax Notation No.1) definiert. Zum Verständnis des SNMP genügen rudimentäre ASN.1-Kenntnisse (siehe Vorlesung).

Diese Abbildung zeigt den Baum unterhalb des Knotens `system`.



Konkrete Werte (Instanzen) werden als Blätter an eine Object Type-Definition in die MIB gehängt. So liefert eine Abfrage auf den Knoten `sysDescr` folgende Antwort:

```
cembalo.wagnerhome [192.168.1.5:161] [Sat Feb 08 01:33:44 CET 2003]:
  sysDescr.0 : Linux cembalo 2.4.18-4GB #1 Wed Mar 27 13:57:05 UTC 2002 i686
```

Konkrete Wertebeispiele für die MIB des UCD-Agenten finden Sie in der zweiten Praktikumsaufgabe.

3 Versuchsaufbau

Die Praktikumsrechner haben eine Linux-Installation. Jeder Rechner im Netz kann sowohl als Manager, als auch als Agent fungieren.

3.1 Der SNMP-Agent

Als Agentensoftware läuft der UCD SNMP-Dämon. Ob er auf Ihrem Rechner wirklich gestartet ist, prüfen Sie mit

```
ps –ef |grep snmpd
```

Konfiguriert wird der Agent über die Datei

```
/etc/ucdsnmpd.conf
```

Die ASN.1-Definitionen der MIB befinden sich im Verzeichnis

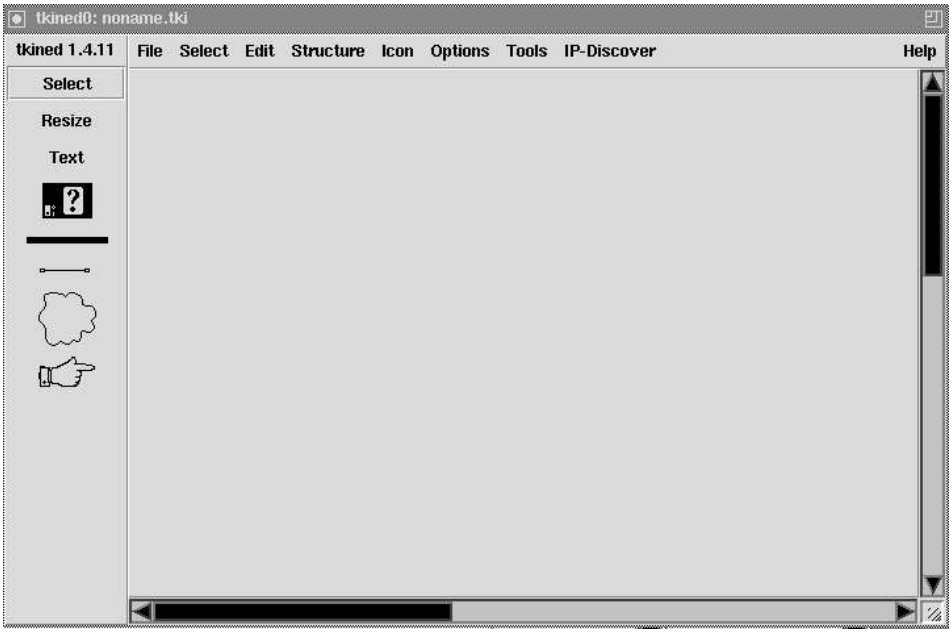
```
/usr/share/snmp/mibs
```

3.2 Der SNMP Manager

Für das Praktikum verwenden wir zwei Arten von Manager-Software:

- interaktiv: tkined (Starten mit `tkined&`)

tkined ist ein graphischer SNMP-Manager. Eine elementare Hilfefunktion finden Sie in den *man pages* (`man tkined`), sowie in den Tool-Menues.



Im Menue „Tools“ können weitere Tool-Menues geöffnet werden. Im Verlauf des Praktikums werden die Tools „IP Discover“, „IP Trouble“, „SNMP Browse“ sowie „SNMP Tree“ benötigt. In dieser Abbildung ist das Menue „IP Discover“ bereits geöffnet.

tkined wird konfiguriert über die Datei

`/usr/lib/tnm2.1.11/library/init.tcl`

Die MIB-Definitionen für den Manager finden sich unter

`/usr/lib/tnm2.1.11/mibs`

- die Kommandozeilenbefehle `snmpget` und `snmpwalk`

`snmpget` liest den Wert eines einzelnen Blattes der MIB. `snmpwalk` liest alle Blätter unter dem angegebenen Knoten der MIB.

`snmpget` und `snmpwalk` gehören auch zum UCD-Paket. Sie referenzieren daher dieselbe MIB wie der UCD-snmpd (`/usr/share/snmp/mibs`).

Syntax:

```
snmpwalk <rechner_name> <domain> <start_knoten>
snmpget  <rechner_name> <domain> <knoten>
```

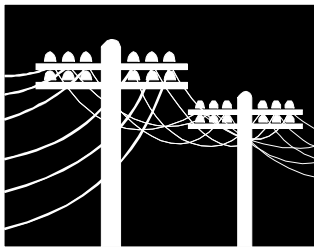
`<rechner_name>` : IP-Adresse/Name des Rechners
`<domain>` : Berechtigungsdomäne. Standardwert: `public`
`<(start)knoten>`: Knoten der MIB in Punktschreibweise `(.1.3.2.4....)`

4 Aufgabenstellung

4.1 Erstellen eines Netzplans

Die Frage, die im ersten Praktikumsschritt beantwortet werden soll, lautet:

„Wie sieht mein Netz aus ?“



Aufgabe:

- Erstellen Sie einen Netzplan (*map*) und drucken Sie diesen aus!

Hinweise zur Ausführung:

Rufen Sie im Menue „IP Discover“ die Funktion „Discover IP Network“ auf. Nachdem Sie im nächsten Dialog die IP-Adresse des Netzwerks eingegeben haben (192.168.1), sucht tkined das lokale Netz ab und versucht eine *map* zu zeichnen.

Danach können Sie das entstandene Bild bearbeiten (Objekte können mit der mittleren Maustaste verschoben werden).

Vergessen Sie nicht das Abspeichern der *map* !

4.2 Auslesen der MIB

Nun sollen konkrete Werte aus der MIB gelesen werden ...

Wählen Sie einen Rechner des Netzwerks an und beantworten Sie folgende Fragen:

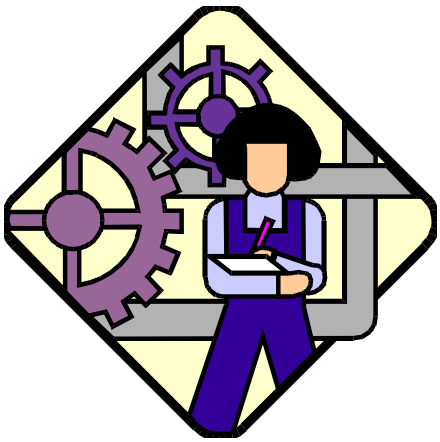
- Welchen Namen und welches Betriebssystem hat der Rechner ? (system)
- Welche Interfaces sind administriert und welche davon sind aktiv ? (interfaces)
- Lesen Sie die IP-Routing-Tabelle des Rechners aus! (ip)
- Wieviele SNMP-Pakete wurden bereits mit dem Rechner, den Sie untersuchen, ausgetauscht ? (snmp)
- Wieviele Festplatten hat der Rechner und wie voll sind diese ? (host)
- Wann und in welcher Version wurde das Softwarepaket „udcsnmpd“ installiert ? (host)

Hinweise zur Ausführung:

Diese Aufgabe können Sie sowohl mit dem tkined-Tool „SNMP Browser“ als auch mit „SNMP Tree“ lösen. Nach jeder Frage ist als Hinweis derjenige Knoten unterhalb „mib-2“ angegeben, unterhalb dessen sich die gesuchten Daten befinden.

4.3 Automatisierte Auswertung der MIB

... doch welcher Administrator sitzt ständig vor dem Rechner ...



Aufgabe:

- Erstellen Sie ein einfaches *shell script*, das periodisch die Prozessorlast eines Rechners überprüft und im Überlastfall eine Warnung ausgibt.

Hinweise zur Ausführung:

Der Lösung dieser Aufgabe kommen Sie mit folgenden Schritten näher:

- Lesen Sie mit dem Kommando `snmpwalk` alle Knoten unter `enterprises.ucdavis.laTable` aus !

- Lesen Sie mit dem Kommando `snmpget` das Fehlerkennzeichen `laErrorFlag` für die über 15 Minuten gemittelte Prozessorlast (Load-15) aus !

- Schreiben Sie ein einfaches *shell script*, das periodisch das Kommnado aus dem letzten Teilschritt ausführt und das Ergebnis mit einem Sollergebnis vergleicht. Im Überlastfall soll eine Meldung ausgegeben werden.

Hier ein Hinweis, wie das Skript aussehen könnte:

```
#!/bin/bash

while [ 1 = 1 ]
do
        checkResult=`snmpget cembalo public
                  enterprises.ucdavis.laTable.laEntry.laErrorFlag.3`
        if [ "$checkResult" != ....
        then
...
...
sleep 10
done
```

4.4 Neue Agentenfunktion

... jetzt bringen wir dem Agenten etwas bei ...

Aufgabe:

- Erweitern Sie die Agentenfunktionalität um eine *set*-Funktion für den Knoten `system.sysLocation` !
- Setzen Sie den Wert für den Knoten auf Ihren Namen !
- Überprüfen Sie das Ergebnis durch die Abfrage dieses Knotens und dokumentieren Sie das Ergebnis durch einen Bildschirmausdruck !

Hinweise zur Ausführung:

Der Knoten `system.sysLocation` beschreibt den Ort, an dem der Rechner steht. Laut MIB-Definition ist er read-write. Daher wird im „SNMP Browser“ bzw. „SNMP Tree“ eine *set*-Funktion angeboten. Versucht man einen Wert zu setzen, führt dies zu einer Fehlermeldung. Dies liegt daran, dass der UCD-Agent die *set*-Funktion nicht unterstützt.

Der UCD-snmpd wird durch die Datei `/etc/ucdsnmpd.conf` gesteuert. In dieser Datei finden wir folgenden Eintrag:

```
syslocation <Ortsangabe>
```

Mit dieser Konfiguration wird der Wert des Knotens festgelegt. Um den Wert änderbar zu machen, müssen wir den Agenten um eine eigene Funktion erweitern. Dazu wird (durch den Betreuer) eine Zeile eingefügt, die angibt, für welche Knoten eine eigene Funktion aufgerufen werden soll:

```
pass <knoten> <funktion>
```

Dabei beschreiben `<knoten>` den Knoten, für den oder unterhalb dem das angegebene Programm `<funktion>` verwendet werden soll.

Beispiel:
```
pass .1.3.6.1.2.1.1.6 /home/student/sysLocationAgent
```

Damit die Änderung gültig wird, muss der snmpd neu gestartet werden.

Das gerufene Programm hat folgende Eigenschaften zu erfüllen:

Aufruftyp	Parameter	Reaktion

snmp-get	`-g .1.3.6.1.2.1.1.6.0`	Ausgabe auf stdout:
		`.1.3.6.1.2.1.1.6.0`
		`string`
		`<Location>`

snmp-get-next	`-n .1.3.6.1.2.1.1.6`	Ausgabe auf stdout:
		`.1.3.6.1.2.1.1.6.0`
		`string`
		`<Location>`

	`-n .1.3.6.1.2.1.1.6.0`	Ausgabe auf stdout: `.1.3.6.1.2.1.1.7`
snmp-set	`-s .1.3.6.1.2.1.1.6.0` `string <new_location>`	Ändern der Datenbasis

In allen anderen Fällen hat das Programm nichts zu tun.

Als Datenbasis verwenden wir eine Datei im Arbeitsverzeichnis.

Als Programmiersprache sei ein bash-Skript empfohlen. Hier nun als Beispiel die Implementierung für dem snmp-get:

```
#!/bin/bash

# build response for get
if [ $1 = "-g" ]
then
     if [ $2 = ".1.3.6.1.2.1.1.6.0" ]  # sysLocation.0
     then
         echo ".1.3.6.1.2.1.1.6.0"
         echo "string"
         cat /home/student/sysLocation.data
     fi
fi
```

5 Literatur

William Stallings, SNMP, SNMPv2, SNMPv3 and RMON 1 and 2
Addison-Wesley, 1999

Bernhard Röhrig, Linux im Netz
C&L Computer und Literaturverlag, 1997