Netzwerkmanagement mit SNMP
→ Teil 2

Prof. Dr. Norbert Pohlmann
Fachbereich Informatik
Verteilte Systeme und Informationssicherheit

Fachhochschule Gelsenkirchen

Inhalt

- **Das Kommunikationsmodell von SNMP**
- **Mängel und Probleme von SNMPv1**

Inhalt

- **Das Kommunikationsmodell von SNMP**
- **Mängel und Probleme von SNMPv1**

Das Simple Network Management Protocol
→ Kommunikationsmodell (1/3)

- Für die Kommunikation zwischen Agent und Manager sind grundsätzlich zwei Vorgehensweisen denkbar:
- **Agent sendet einen Trap**
  - Im Fall von **ungewöhnlichen Ereignissen**, z.B. Abbuch einer Verbindung oder Überschreiten eines Grenzwertes, sendet der verwaltete Knoten einen **Interrupt** (**Trap** genannt) an die Managementstation.
  - Dies hat den Vorteil, dass bei Auftreten eines **Störfalles** die Managementstation sofort verständigt wird.
  - Nachteile dieser Vorgehensweise sind, dass zusätzliche Betriebsmittel auf den verwalteten Knoten benötigt werden, z.B. CPU-Zeiten oder Speicher, und dass bei häufigem Auftreten solcher Ereignisse ein zusätzlicher, von der Managementstation **nicht kontrollierbarer** Netzverkehr, auftritt.
  - Dies kann unerwünschte Folgen haben, wenn z.B. in einem überlasteten Netz **sämtliche Knoten die Überlastung des Netzes der Managementstation mitteilen** und dadurch zusätzlichen Netzverkehr verursachen.

Das Simple Network Management Protocol
→ Kommunikationsmodell (2/3)

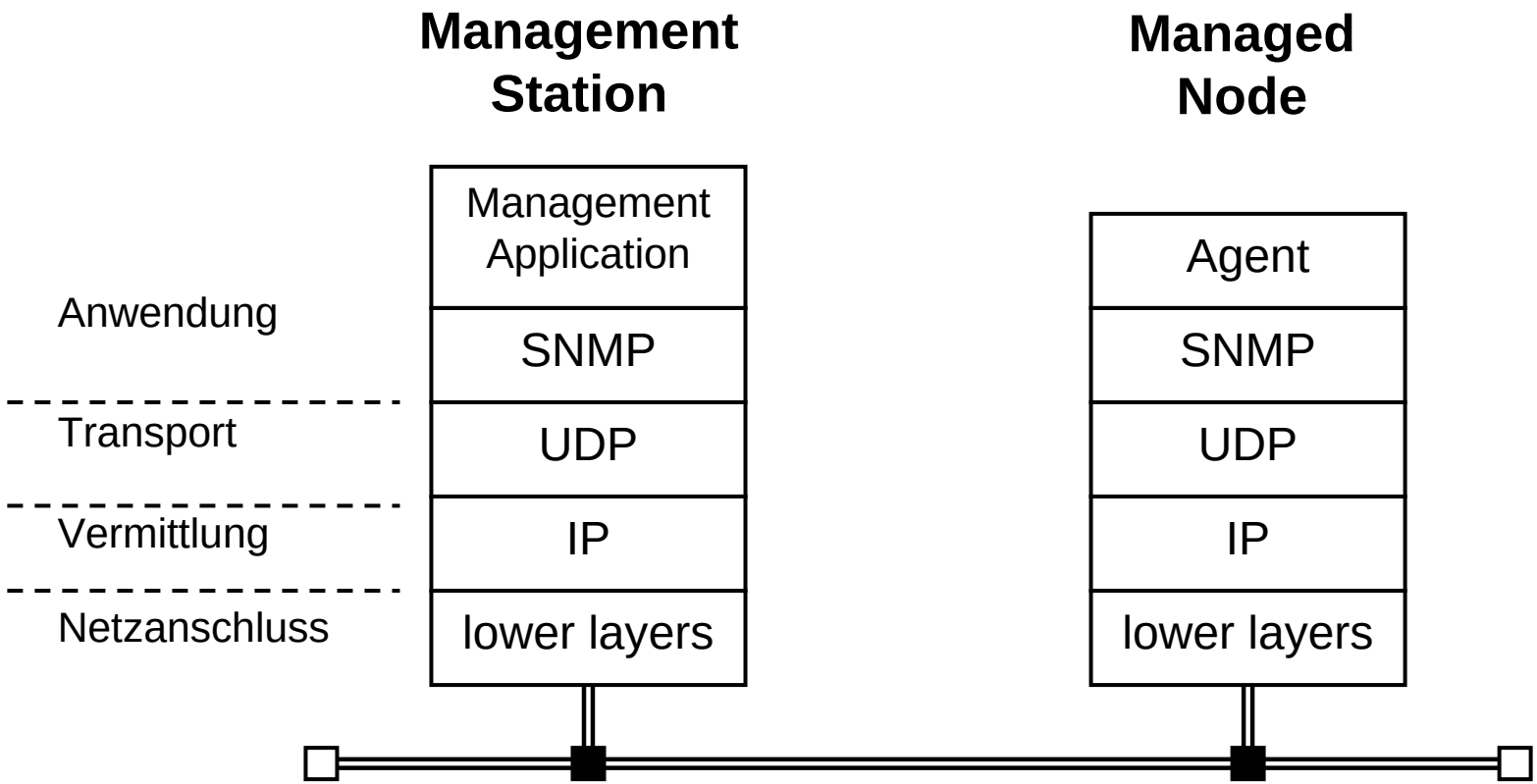
- **Polling der Agenten**
  - Die **Managementstation fragt periodisch** die zu verwaltenden Knoten ab, ob alles in Ordnung ist.
  - Dies hat gegenüber Traps zum einen den Vorteil, dass Polling **sehr einfach zu realisieren ist**.
  - Zum anderen hat der Netzverwalter die Möglichkeit, den zusätzlichen Managementverkehr zu kontrollieren, indem er das **Pollinginterval** entsprechend **variiert**.
  - Der entscheidende Nachteil am Polling ist, dass die Managementstation (bzw. der Netzverwalter) nicht wissen kann, **wann** sie welche Konten abfragen und in **welchen Abständen diese Abfrage** erfolgen soll.
  - Ist der Abstand **zu klein**, so wird **Übertragungskapazität verschenkt**; ist er **zu groß**, kann die **Reaktionszeit auf einen Störfall zu groß** sein.

Das Simple Network Management Protocol
→ Kommunikationsmodell (3/3)

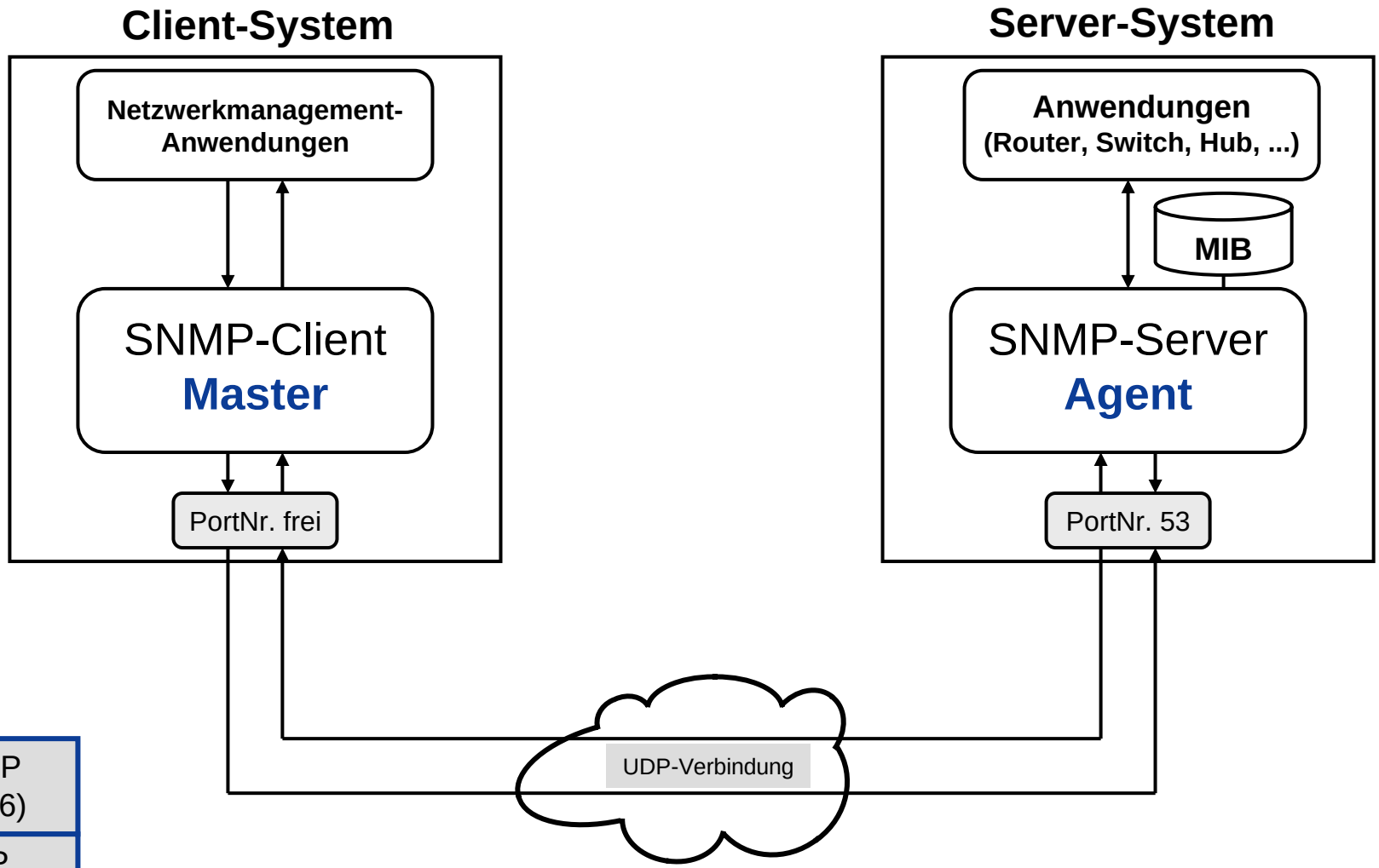
- **Trap-directed Polling**
  - Im Internet-Standard Networkmanagement Framwork wird das Modell des **trap-directed polling** (Traps beeinflussen das Polling) benutzt.
  - Falls ein besonders Ereignis eintritt, schickt der verwaltete Knoten einen **einzigen einfachen Trap** an die Managementstation.
  - Die Managementstation ist dann dafür verantwortlich, durch weitere Anfragen an den verwalteten Knoten, die Art und das Ausmaß des Problems zu bestimmen.
  - Dieser Kompromiss ist **erstaunlich wirkungsvoll**: **der Einfluß auf die verwalteten Knoten bleibt klein**, die **Auswirkungen auf die Bandbreite des Netzes werden minimiert**, und bei Problemen kann dennoch rechtzeitig reagiert werden.
  - SNMP verwendet das verbindungslose Transport-Protokoll, das keine gesicherte Übertragung der Daten gewährleistet.

Einordnung von SNMP in das TCP/IP-Sichtenmodell

Das *Simple Network Management Protocol* ist im Internet Schichten-Modell in der **Applikationsschicht** angesiedelt.

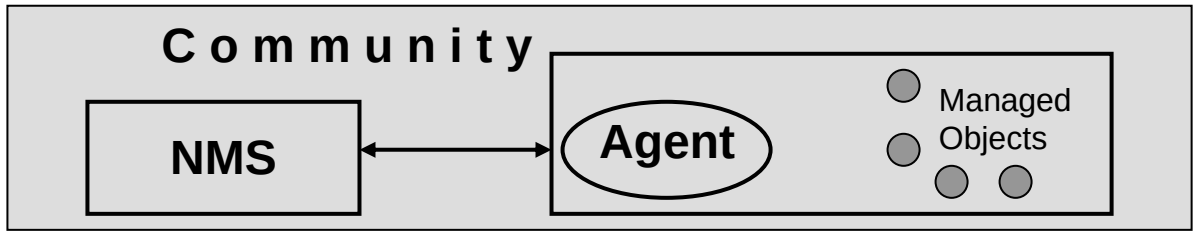


SNMP
→ Client-Server Beziehung



SNMP (OSI 6)
UDP
IP

Ausweiskontrolle und Zugriffskontrolle (1/2)
→ Community



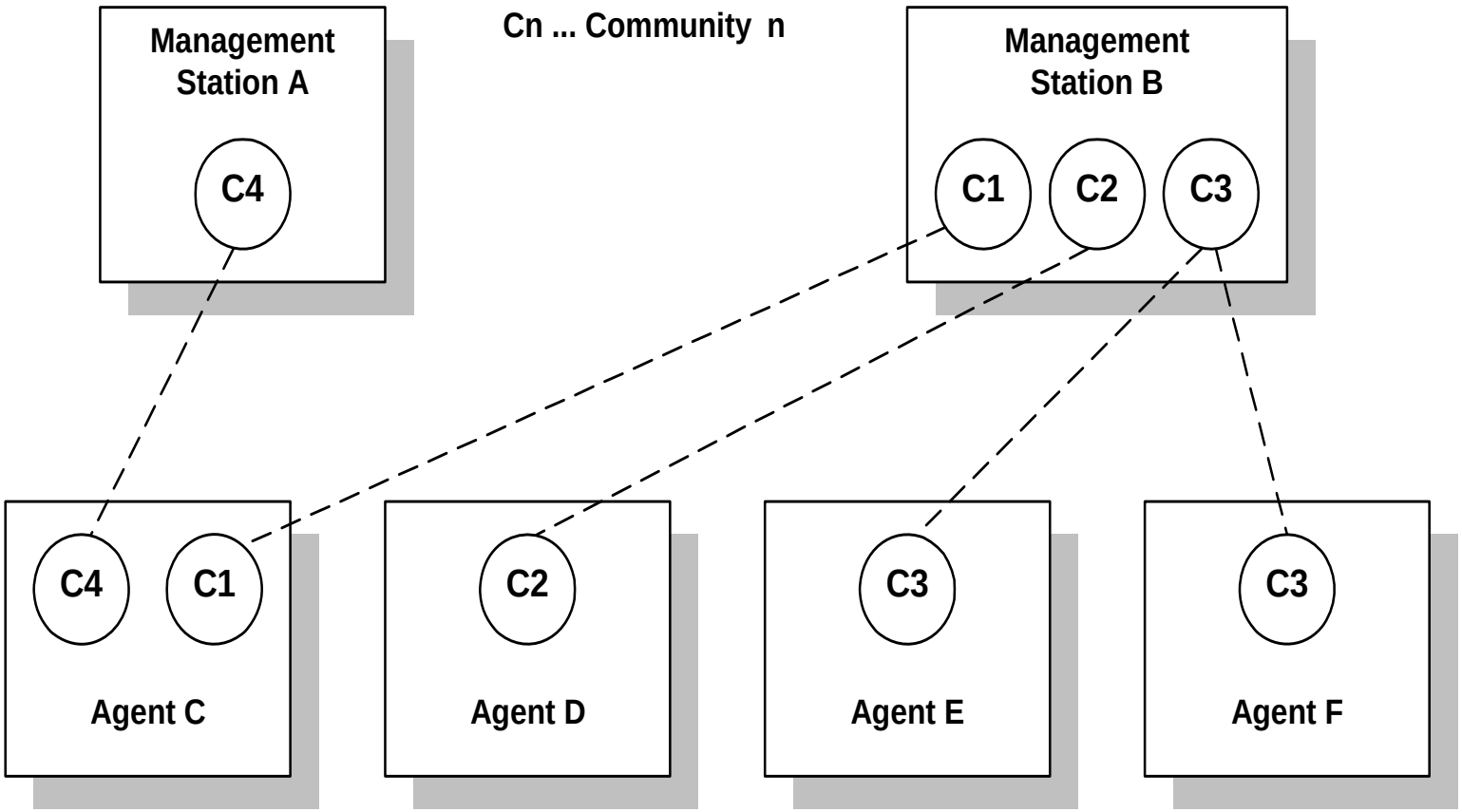
- Die Ausweis- und Zugriffskontrolle in SNMP basiert auf dem **Community Konzept**.
- Als Community wird die Verbindung eines SNMP-Agenten mit einer oder mehreren Managementstationen bezeichnet.
- Eine Community wird von dem Agenten (und nicht von der Managementstation) festgelegt und besitzt einen innerhalb des Agenten eindeutigen **Community-Namen**.
- Ein Agent kann auch mehrere Communities definieren.
- So definiert in der Regel jeder Agent eine Community **„public“,** die einen **beschränkten Zugriff auf die Managed Objects eines Agenten erlaubt**.

Ausweiskontrolle und Zugriffskontrolle (2/2)
→ Aufbau einer SNMP-Nachricht

Versions-nummer	Community Namen	Daten (PDU)

- Wenn SNMP-Einheiten Nachrichten austauschen, so enthalten diese grundsätzlich immer drei Anteile:
  - Das erste Feld jeder SNMP-Nachricht enthält eine SNMP-Versionsnummer.
    Für die erste Version von SNMP ist der Wert „0“ definiert.
    (SNMPv1 = 0)
  - Danach folgt der Community-Name als Nachweis, dass die gesendete SNMP-Einheit Mitglied der angegebenen Community ist und
  - der Datenbereich, der eine SNMP-PDU enthält.

SNMP-Zugriffsprinzip (community names)



Ausweiskontrolle

- In SNMPv1 ist nur eine triviale Ausweiskontrolle vorhanden, indem der ***Community-Name*** einfach als **Passwort** verwendet wird.
- Erschwerend kommt hinzu, dass der ***Community-Name*** in der SNMP-Nachricht **unverschlüsselt** übertragen wird.
- Falls der *Community-Name* der empfangenden SNMP-Einheit bekannt ist, gilt die sendende SNMP-Einheit als Mitglied dieser Community und die empfangene Anfrage wird bearbeitet.
- Im anderen Fall tritt ein Fehler bei der Ausweiskontrolle auf, und die empfangende SNMP-Einheit kann abhängig von ihrer Voreinstellung einen authenticationFailure **Trap** erzeugen.

Zugriffskontrolle

- Sobald die sendende SNMP-Einheit sich als Mitglied der *Community* ausgewiesen hat, muss der Agent festlegen, welche Zugriffe erlaubt sind.
- Dabei setzt sich die Zugriffskontrolle aus zwei Aspekten zusammen:
  - **MIB-View:** Unter dem Begriff MIB-View werden alle innerhalb einer MIB sichtbaren *Managed Objects* zusammengefasst.
    Für jede *Community* können unterschiedliche Sichtweisen auf der MIB eines Agenten implementiert werden.
  - **SNMP Zugriffsmodus:** Für jede *Community* wird ein Zugriffsmodus (read-only oder read-write) definiert.
- Die Kombination aus MIB-View und SNMP Zugriffsmodus wird auch als ***SNMP Community Profile*** bezeichnet.
- Ein *Community Profile* besteht damit aus einer definierten Teilmenge der MIB eines Agenten und einem Zugriffsmodus für Objekte dieser Teilmenge.

Schnittstelle aus SNMP- und MIB-Zugriffsmodus

Zugriffsmodus der Community	Objekt-Zugriff entsprechend der MIB			
	read-only	read-write	write-only	not accessable
read-only	get, get-next, trap	get, get-next, trap	---	---
read-write	get, get-next, trap	get, get-next, set, trap	set, trap	---

- Bei der Definition einer MIB wird für jedes Objekt ein Zugriffsrecht festgelegt.
- Aus der Schnittmenge der Zugriffsmodi für die Managed Objects und Community ergeben sich damit neue Zugriffsrechte für die einzelnen Objekte.
- Mit dem Community Konzept ist es möglich, dass ein Anwender - je nach verwendetem Community-Name - unterschiedliche Sichten und Rechte für die einzelnen Objekte einer MIB besitzt.

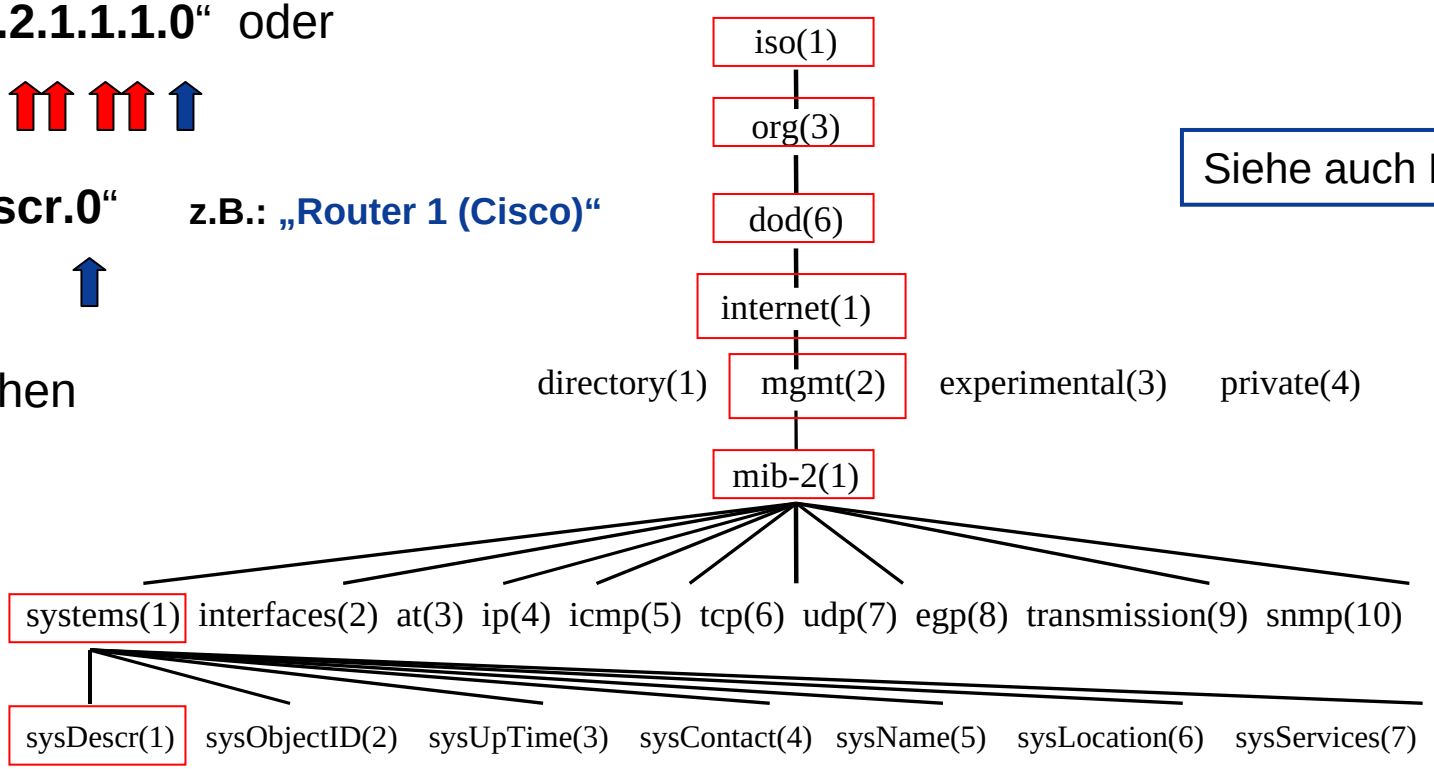
Zugriff auf Objektinstanzen

- Das Ziel von SNMP besteht darin, einem Manager Zugriff auf die *Managed Objects* eines Agenten bereitzustellen.
- Hierzu muss festgelegt werden, wie die betroffenen Objekte innerhalb einer SNMP-Operation angesprochen werden.
- **SNMP beschränkt den Zugriff auf einzelne *Managed Objects*** (Blätter in der MIB-Baustruktur), so ist es z.B. nicht möglich, direkt auf eine ganze Tabelle oder Zeilen von Tabellen zuzugreifen.

Zugriff auf einfache *Managed Objects*

- Im SNMP-Protokoll wird vereinbart, dass auf einfache Variablen (die nicht Teil einer Tabelle sind) durch Anhängen einer **„0“** an den *Object Identifier* zugegriffen wird.
- Damit wird beispielsweise in einer SNMP-Operation die MIB-Variable zur Beschreibung des Netzwerkgerätes durch den Zugriffsidentifikator
  - „**1.3.6.1.2.1.1.1.0**“ oder
  - „**sysDescr.0**“ **z.B.: „Router 1 (Cisco)“**

angesprochen



Siehe auch PM

Zugriff auf Spaltenobjekte (1/2)

- Für *Managed Objects* die Teil einer Tabelle sind, ist der Instanzenzugriff komplexer.
- Die Tabellenstruktur führt dazu, dass jetzt in jeder Tabellenzeile gewissermaßen eine Instanz des Objekts existiert.
- Die Aufgabe besteht darin, neben der Tabellenspalte, die durch den *Objekt Identifier* eindeutig bestimmt ist, zusätzlich die Tabellenzeile anzugeben, um die entsprechende Variable eindeutig zu identifizieren.
- Um dies zu erreichen, muss jede Tabelle ein Index-Spaltenobjekt besitzen, dessen Wert innerhalb der Tabelle eindeutig ist und somit eine Zeile der Tabelle eindeutig identifiziert.
- Dieses Objekt wird in der ASN.1 Beschreibung der MIB mit dem Schlüsselwort **INDEX** markiert.

Zugriff auf Spaltenobjekte (2/2)

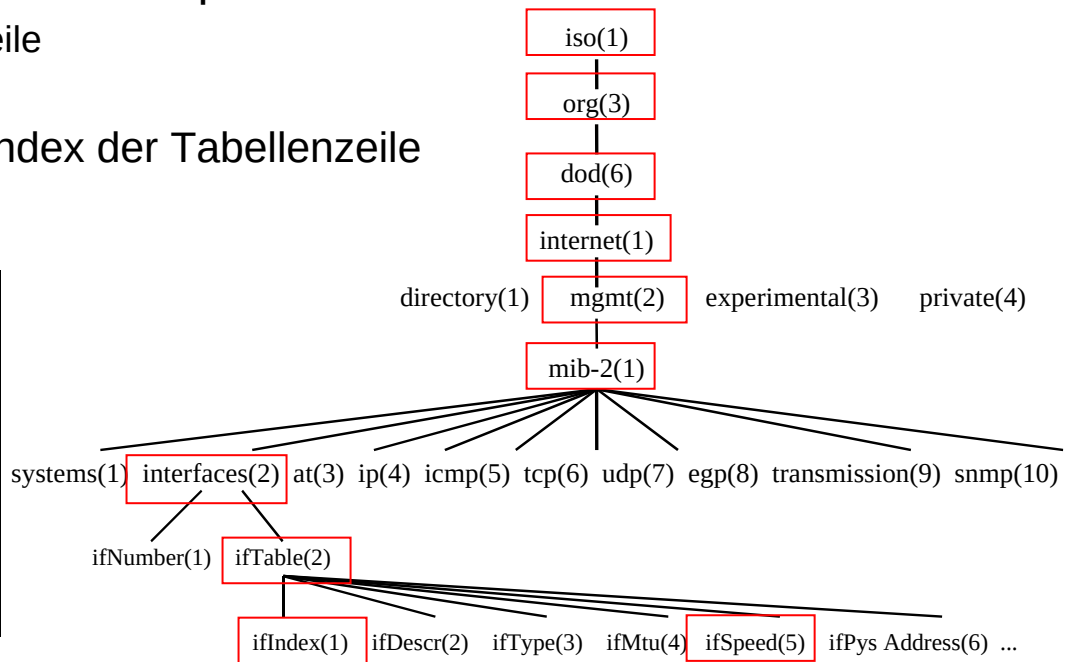
- Soll auf ein Objekt innerhalb einer Tabelle zugegriffen werden, so muss der *Objekt Identifier* des betreffenden Spaltenobjektes und der Wert des Indexobjektes zur Identifikation angegeben werden.
- Um beispielsweise in der Schnittstellentabelle ifTable die Übertragungsgeschwindigkeit des dritten Interfaces zu bekommen, wird der folgende Zugriffsidentifikator verwendet:
  - „1.3.6.1.2.1.2.2.1.5.3“ oder ifSpeed.3

    Tabellenspalte I Tabellenzeile

    = *Object Identifier* von ifSpeed = Index der Tabellenzeile

Interfacetabelle: ifTable

Objekttype	ifIndex	. . .	ifSpeed	. . .	ifSpecific
Feld Nr.	1	. . .	5	. . .	22
Interface Nr.					
1	1		9600		0
2	2		64000		0
3	3		1200		0
.					
.					
.					



Operationen von SNMP (1/2)

- Das SNMP-Protokoll ist eine asynchrones Frage/Antwort Protokoll.
- Eine SNMP-Einheit muss nicht auf eine Antwort warten, nachdem sie eine Nachricht gesendet hat.
- Sie kann weitere Nachrichten senden oder andere Dinge tun.
- Des weiteren liegt es an der sendenden SNMP-Einheit, die gewünschte Zuverlässigkeit zu implementieren, da Fragen oder Anwort durch den darunterliegenden Transportdienst verloren gehen können.
- Insgesamt stehen vier einfache SNMP Befehle zu Verfügung:
  - **get**
  - **get-next**
  - **set**
  - **trap**

Operationen von SNMP (2/2)

- Alle vier Befehle erlauben den Zugriff auf mehrere einzelne Objekte in einem einzigen SNMP-Befehl.
- Dazu enthält jeder SNMP Befehl ein VarBindList Feld, das folgendermaßen in ASN.1 definiert ist:

- ```
  VarBindList   ::=     SEQUENCE OF VarBind

  VarBind       ::=     SEQUENCE      {
                            name Objectname,
                            value ObjectSyntax
                                  }
  ```

- Diese Feld enthält damit eine Liste von *Managed Objects* repräsentiert durch den jeweiligen *Object Identifier* und dem eigentlichen Wert des Objektes.
  ```

Der get Operator (1/2)

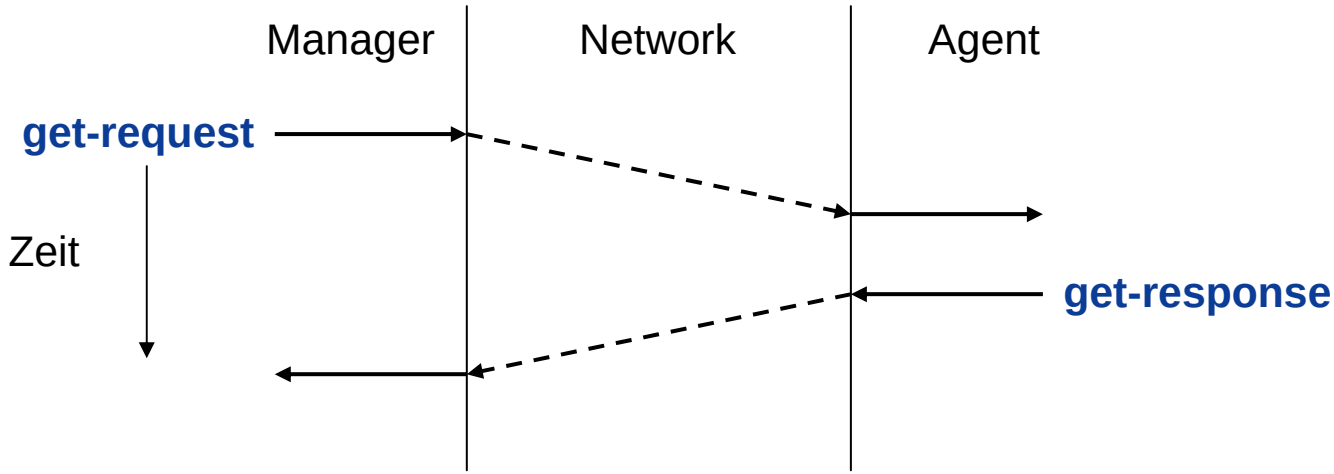
- Mit Hilfe des get Operator kann der Manager Informationen vom Agenten einholen.
- Um beispielsweise die Systembeschreibung eines Netzelementes auszulesen, wird der folgende Befehl an den Agenten gesendet.

  get (sysDescr.0, < >)

- Als Antwort auf den get Befehl sendet der Agent eine GET-RESPONSE PDU, die exakt denselben Aufbau besitzt, bei der aber die Wertefelder innerhalb des VarBindList Feldes entsprechend ausgefüllt sind:

  get-response (sysDescr.0, <„Router for building ...“>)

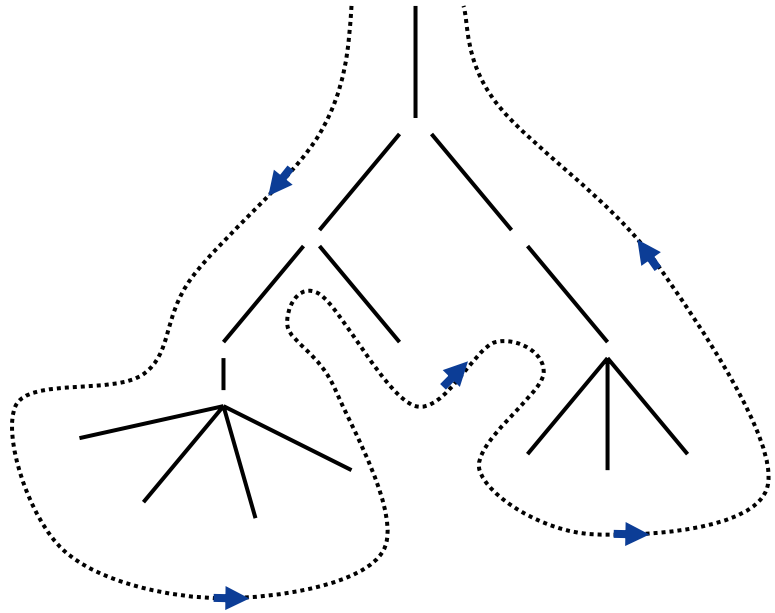
Der get Operator (2/2)



- Der get Operator ist **atomar.** Entweder werden die Werte aller abzufragenden Objekte zurückgegeben oder überhaupt keine.
- Existiert zum Beispiel eines der Objekte indem VarBindList Feld nicht, so wird die Bearbeitung abgebrochen und der Fehler noSuchName zurückgeliefert.
- Dies hat zur Folge, dass auch die Werte der anderen Parameter verloren gehen.

Der get-next Operator (1/3)

- Der get-next Operator liest nicht den Wert der angegebenen Instanz, sondern den Wert der in der MIB-Hierachie folgenden Instanz aus.
- Die Reihenfolge der *Managed Objects* innerhalb der MIB Baumstruktur lässt sich am einfachsten ermitteln, indem man von links beginnend am MIB Baum entlangfährt:



Der get-next Operator (2/3)

- Um beispielsweise die Systembeschreibung eines Agent, könnte der Manager einen der folgenden get-next Befehle an den Agenten senden.

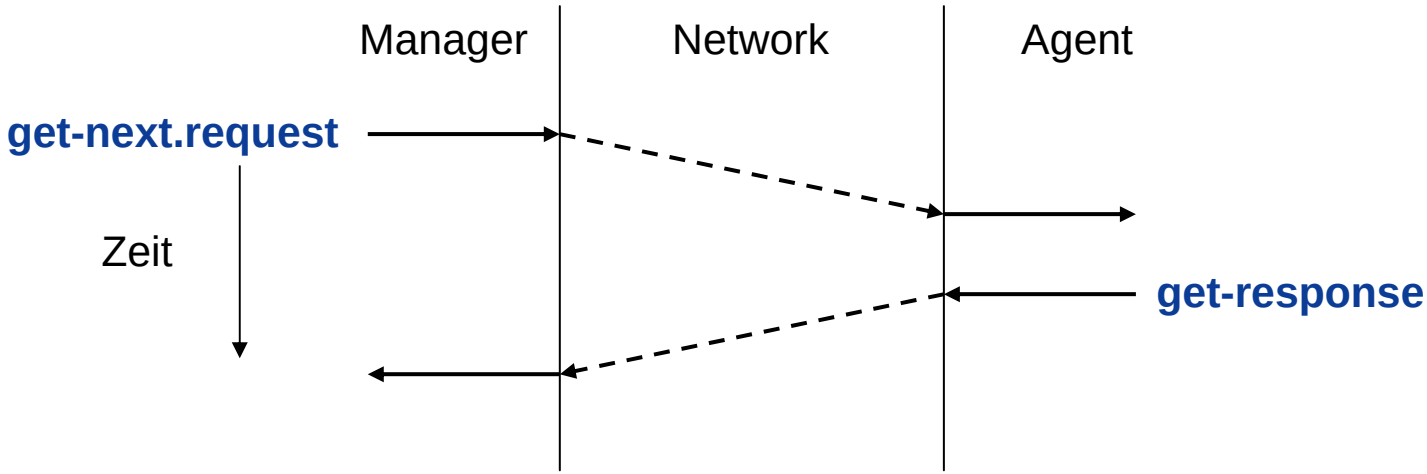
  get-next (sysDescr, < >) oder get-next (system, < >)

- Der Agent sucht daraufhin nach der nächsten Objektinstanz in seiner lokalen MIB.
- In beiden Fällen ist die nächste Instanz „sysDescr.0“, so dass der Agent beide Befehle mit der folgenden GET-RESPONSE PDU beantwortet:

  get-response (sysDescr.0, <„Router for building ...“>)

- Es wird nicht nur der entsprechende Wert von sysDescr zurückgeliefert, sondern auch der entsprechende *Object Identifier*.
- Möchte der Manager nun die gesamte system Gruppe auslesen, so sendet er einfach mehrere get-next Befehle, wobei der Wert des *Object Identifiers* vom **vorherigen Befehl als Eingabe** des **nächsten Befehls verwendet** wird.

Der get-next Operator (3/3)



- Einer der Vorteile des get-next Operators gegenüber dem einfachen get-Befehl ist, dass bei Bearbeitung mehrerer übergebener Parameter nicht mit einem Fehler abgebrochen wird, sondern - falls einer der Parameter nicht existieren sollte - einfach der Wert der nachfolgenden Instanz übergeben wird.
- Der Manager muss jedoch den zurückgelieferten OID überprüfen, ob auch der Wert der gewünschten Variable zurückgeliefert wird.

Der set Operator (1/2)

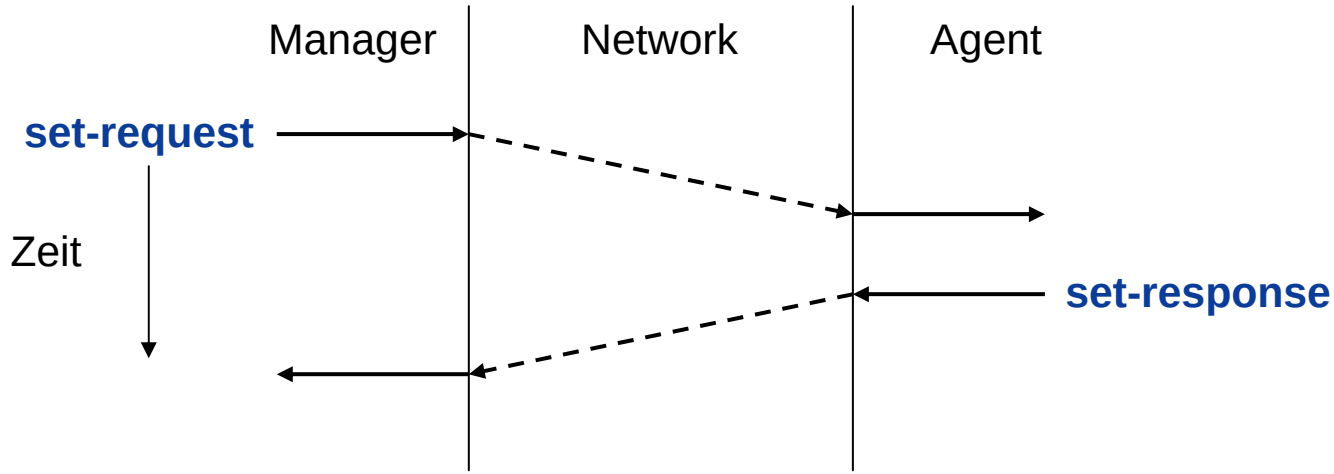
- Mit dem set Operator kann ein Manager MIB-Variablen Werte zuweisen.
- Mit dem Setzen von MIB-Variablen können auch bestimmte Aktionen angestoßen werden (z.B. das Initialisieren oder Testen von Netzkomponenten).
- Damit besteht die Möglichkeit, das Netz nicht nur zu überwachen, sondern in einem gewissen Maße auch steuern zu können.
- Um beispielsweise die Systembeschreibung für einen Netzknoten zu ändern, kann der folgenden set Befehl verwendet werden.

  set (sysDescr.0, <„Printer for department ... >)

- Der Agent übermittelt in einer SET-RESPONSE PDU den Erfolg oder Misserfolg der Schreibaktion:

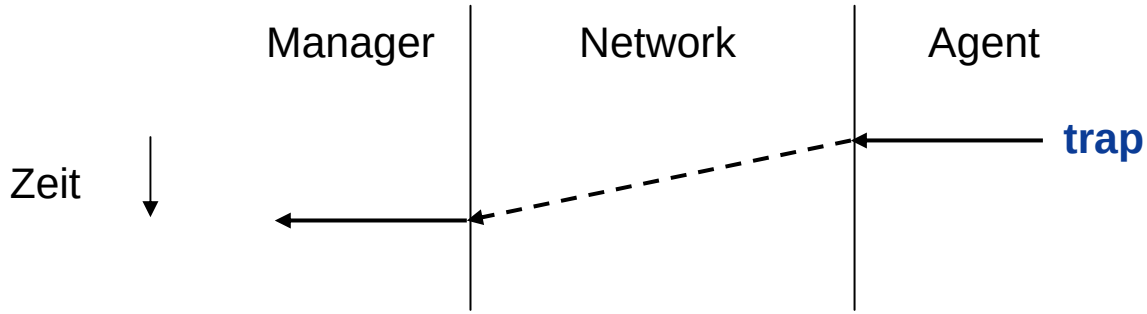
  set -response (sysDescr.0, <„Printer for department ... >)

Der set Operator (2/2)



- Der set Operator ist **atomar**. Entweder es können alle Werte innerhalb eines Agent gesetzt werden oder es werden überhaupt keine Werte geändert.
- Falls der Agent aus irgendeinem Grund den Wert eines *Managed Objects* nicht ändern kann, so liefert er einen entsprechenden Fehlerstatus an den Manager zurück.
- Ein Manager kann nur dann den Wert eines Objektes setzen, falls sowohl die *Community* als auch das Objekt selbst den Zugriffsmodus „read-write“ besitzt.

Der trap Operator



- Erkennt der Agent eine besondere Situation, so sucht der Agent jene Manager heraus, denen er sogenannte **Traps** schicken soll.
- Für jeden Manager wählt er dazu eine passende *Community* und sendet ein TRAP-PDU zu diesem Manager.
- Innerhalb des VarBindList Feldes kann der Agent zusätzliche Informationen an den Manager senden.
- Empfängt ein Manager eine Trap-PDU, so stellt er dieses Ereignis entsprechend dar, um die Aufmerksamkeit des Anwenders darauf zu lenken.
- Bei einem Trap des Agenten erfolgt keine Bestätigung durch die Managementstation.

Aufbau der SNMP-Pakete
→ PDU der Befehle GET, GET-NEXT und SET (1/4)

request-id	error-status	error-index	varBindList

- **request-id**
  - Ist ein **ganzzahliger Wert**, der vom Manager dazu benutzt wird, zwischen noch ausstehenden Anfragen zu unterscheiden.
  - Damit kann eine Verwaltungsanwendung, falls sie es möchte, schnell mehrere SNMP-Nachrichten hintereinander schicken.
  - Die Antworten können anhand der request-id den entsprechenden Operationen zugeordnet werden.

Aufbau der SNMP-Pakete
→ PDU der Befehle GET, GET-NEXT und SET (2/4)

request-id	error-status	error-index	varBindList

- **error-status**
  - Falls der Status nicht Null ist, zeigt er an, dass eine Ausnahme bei der Bearbeitung auftrat. Die möglichen Werte sind:
    - **tooBig**, der Agent konnte das Ergebnis nicht in eine einzige SNMP-Nachricht packen.
    - **noSuchName**, die gewünschte Operation bezog sich auf einen unbekannten Variablennamen.
    - **badValue**, die gewünschte Operation gab beim Versuch, eine Variable zu ändern, eine flasche Syntax oder einen falschen Wert an.
    - **readOnly**, die gewünschte Operation versuchte, eine Variable zu verändern, die nicht beschrieben werden darf.

Aufbau der SNMP-Pakete
→ PDU der Befehle GET, GET-NEXT und SET (3/4)

request-id	error-status	error-index	varBindList

- **error-index**
  - Ist diese Feld nicht Null, so zeigt es auf die Variable in der Anfrage, die fehlerhaft war.
  - Diese Feld ist nur für die Fehler noSuchName, badValue und readOnly ungleich Null.
  - In diesen Fällen enthält error-index den Abstand innerhalb des varBindList Feldes.
  - Die erste Variable innerhalb dieses varBindList Feldes besitzt dabei den Abstand 1.

Aufbau der SNMP-Pakete
→ PDU der Befehle GET, GET-NEXT und SET (4/4)

request-id	error-status	error-index	varBindList

- **varBindList**
  - Ist eine Liste von Variablen mit je einem **Namen und Wert.**
  - Für die Datentypen GetRequest-PDU und GetNextRequest-PDU ist der Werteteil einer Variablen ohne Bedeutung.
  - Als Konvention gilt hier, dass der Wert immer eine Instanz des ASN.1-Datentyps NULL ist.
  - Diese Felder werden dann von den Agent mit den entsprechenden Werten aufgefüllt und als Response-PDUen an die Managementstation zurückgesendet.

Aufbau der SNMP-Pakete
→ PDU von Traps (1/4)

enterprise	agent-addr	generic-trap	specific-trap	time-stap	

- **enterprise**
  - Enthält den Wert der MIB-Variablen sysObjectID des Agenten und identifiziert die Software des Agenten.
- **agent-addr**
  - Enthält die Netzwerkadresse des Agenten.

Aufbau der SNMP-Pakete
→ PDU von Traps (2/4)

enterprise	agent-addr	generic-trap	specific-trap	time-stap	variable-bindings

- **generic-trap (1/2)**
  - **coldstart**, der Agent initialisiert sich (wieder) selbst, und die Objekte in seinem View können sich ändern.
  - **warmStart**, der Agent initialisiert sich (wieder) selbst, aber die Objekte in seinem View werden nicht verändert.
  - **linkdown**, eine angeschlossene Schnittstelle hat ihren Zustand von up auf down verändert. Die Schnittstelle wir dabei in der ersten Variablen innerhalb des variable-bindings Feldes angegeben.
  - **linkup**, eine angeschlossene Schnittstelle hat ihren Zustand von down auf up verändert. Die Schnittstelle wir dabei in der ersten Variablen innerhalb des variable-bindings Feldes angegeben.

Aufbau der SNMP-Pakete
→ PDU von Traps (3/4)

enterprise	agent-addr	generic-trap	specific-trap	time-stap	variable-bindings

- **generic-trap (2/2)**
  - **authenticationFailure**, eine SNMP-Nachricht wurde von einer SNMP-Einheit empfangen, die fälschlicherweise behauptete, Mitglied einer bestimmten Community zu sein.
  - **egpNeighborLoss**, ein EGP-Partner (Gateway) ist in den Zusand down gewechselt. Die erste Variable im variable-bindings Feldes bezeichnet dabei die IP-Adresse des EGP-Partners.
  - **enterpriseSpecific**, ein anderes unerwartetes Ereignis ist eingetreten. Es wird im specific-trap Feld genauer bezeichnet.

Aufbau der SNMP-Pakete
→ PDU von Traps (4/4)

enterprise	agent-addr	generic-trap	specific-trap	time-stap	variable-bindings

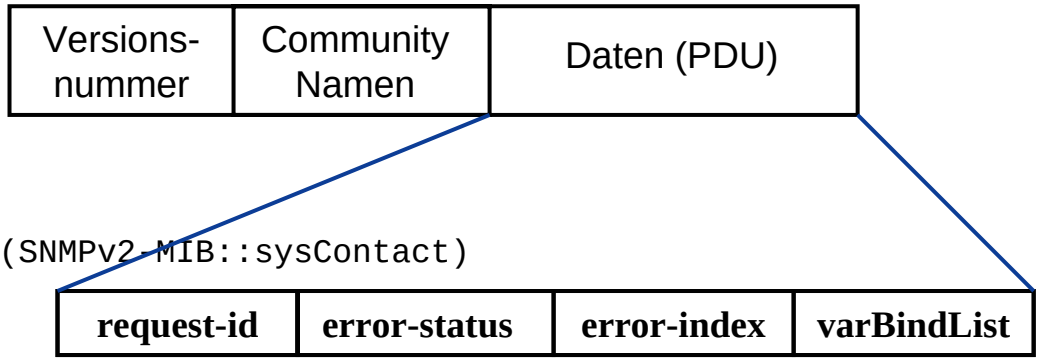
- **specific-trap**
  - Bezeichnet den enterpriseSpecific-Trap, der sich ereignete. Ansonsten ist dieser Wert Null.
- **time-stamp**
  - Enthält den Wert des MIB-Objektes sysUpTime des Agenten, zu dem Zeitpunkt als dieses Ereignis auftrat.
- **variable-bindings**
  - Eine Liste von Variablen, die weitere Informationen über den Trap beinhalten.

SNMP - Protokollmitschnitte

SNMP
→ Protokollmittschnitt - GET

```
No.  Time        Source          Destination     Protocol  Info
  1  0.000000    172.16.16.8     172.16.0.30     SNMP      GET SNMPv2-MIB::sysContact
  2  0.016335    172.16.0.30     172.16.16.8     SNMP      RESPONSE SNMPv2-MIB::sysContact
```

```
Frame 1 (84 bytes on wire, 84 bytes captured)
Internet Protocol, Src Addr: 172.16.16.8 (172.16.16.8), Dst Addr: 172.16.0.30 (172.16.0.30)
User Datagram Protocol, Src Port: 32817 (32817), Dst Port: snmp (161)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET (0)
    Request Id: 0x5e79d13b
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4 (SNMPv2-MIB::sysContact)
    Value: NULL
```




```
Frame 2 (88 bytes on wire, 88 bytes captured)
Internet Protocol, Src Addr: 172.16.0.30 (172.16.0.30), Dst Addr: 172.16.16.8 (172.16.16.8)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 32817 (32817)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x5e79d13b
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4 (SNMPv2-MIB::sysContact)
    Value: STRING: 3Com
```

SNMP
→ Protokollmittschnitt - SET

```
No. Time        Source        Destination   Protocol Info
  1 0.000000    172.16.16.8   172.16.0.30   SNMP     SET SNMPv2-MIB::sysContact.0
  2 0.047465    172.16.0.30   172.16.16.8   SNMP     RESPONSE SNMPv2-MIB::sysContact.0

Frame 1 (125 bytes on wire, 125 bytes captured)
Internet Protocol, Src Addr: 172.16.16.8 (172.16.16.8), Dst Addr: 172.16.0.30 (172.16.0.30)
User Datagram Protocol, Src Port: 32817 (32817), Dst Port: snmp (161)
Simple Network Management Protocol
    Version: 1 (0)
    Community: manager
    PDU type: SET (3)
    Request Id: 0x0dbc14a7
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: STRING: netadmin@informatik.fh-gelsenkirchen.de

Frame 2 (125 bytes on wire, 125 bytes captured)
Internet Protocol, Src Addr: 172.16.0.30 (172.16.0.30), Dst Addr: 172.16.16.8 (172.16.16.8)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 32817 (32817)
Simple Network Management Protocol
    Version: 1 (0)
    Community: manager
    PDU type: RESPONSE (2)
    Request Id: 0x0dbc14a7
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: STRING: netadmin@informatik.fh-gelsenkirchen.de
```

SNMP
→ Protokollmittschnitt - GET NEXT: Übersicht

No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::system
2	0.018269	172.16.0.30	172.16.16.8	SNMP	RESPONSE SNMPv2-MIB::**sysDescr.0**
3	0.018652	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::**sysDescr.0**
4	0.037035	172.16.0.30	172.16.16.8	SNMP	RESPONSE SNMPv2-MIB::**sysObjectID.0**
5	0.051330	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::**sysObjectID.0**
6	0.067624	172.16.0.30	172.16.16.8	SNMP	RESPONSE SNMPv2-MIB::**sysUpTime.0**
7	0.080811	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::**sysUpTime.0**
8	0.098810	172.16.0.30	172.16.16.8	SNMP	RESPONSE SNMPv2-MIB::**sysContact.0**
9	0.116780	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::**sysContact.0**
10	0.132295	172.16.0.30	172.16.16.8	SNMP	RESPONSE SNMPv2-MIB::**sysName.0**
11	0.151317	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::**sysName.0**
12	0.168927	172.16.0.30	172.16.16.8	SNMP	RESPONSE SNMPv2-MIB::**sysLocation.0**
13	0.180550	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::**sysLocation.0**
14	0.198404	172.16.0.30	172.16.16.8	SNMP	RESPONSE SNMPv2-MIB::**sysServices.0**
15	0.222329	172.16.16.8	172.16.0.30	SNMP	GET-NEXT SNMPv2-MIB::**sysServices.0**
16	0.237332	172.16.0.30	172.16.16.8	SNMP	RESPONSE IF-MIB::ifNumber.0

SNMP
→ Protokollmittschnitt - GET NEXT: Übersicht

```
Frame 1 (83 bytes on wire, 83 bytes captured)
Internet Protocol, Src Addr: 172.16.16.8 (172.16.16.8), Dst Addr: 172.16.0.30 (172.16.0.30)
User Datagram Protocol, Src Port: 32817 (32817), Dst Port: snmp (161)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x1dedc309
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1 (SNMPv2-MIB::system)
    Value: NULL
```

```
Frame 2 (134 bytes on wire, 134 bytes captured)
Internet Protocol, Src Addr: 172.16.0.30 (172.16.0.30), Dst Addr: 172.16.16.8 (172.16.16.8)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 32817 (32817)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x1dedc309
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-MIB::sysDescr.0)
    Value: STRING: 3Com SuperStackII Switch 3000 FX, SW Version:2.10
```

SNMP
→ Protokollmittschnitt - GET NEXT: Übersicht

```
Frame 3 (85 bytes on wire, 85 bytes captured)
Internet Protocol, Src Addr: 172.16.16.8 (172.16.16.8), Dst Addr: 172.16.0.30 (172.16.0.30)
User Datagram Protocol, Src Port: 32817 (32817), Dst Port: snmp (161)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x1dedc30a
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-MIB::sysDescr.0)
    Value: NULL
```

```
Frame 4 (94 bytes on wire, 94 bytes captured)
Internet Protocol, Src Addr: 172.16.0.30 (172.16.0.30), Dst Addr: 172.16.16.8 (172.16.16.8)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 32817 (32817)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x1dedc30a
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.2.0 (SNMPv2-MIB::sysObjectID.0)
    Value: OID: SNMPv2-SMI::enterprises.43.1.8.22
```

SNMP
→ Protokollmittschnitt - GET NEXT: Übersicht

```
Frame 5 (85 bytes on wire, 85 bytes captured)
Internet Protocol, Src Addr: 172.16.16.8 (172.16.16.8), Dst Addr: 172.16.0.30 (172.16.0.30)
User Datagram Protocol, Src Port: 32817 (32817), Dst Port: snmp (161)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x1dedc30b
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.2.0 (SNMPv2-MIB::sysObjectID.0)
    Value: NULL
```

```
Frame 6 (89 bytes on wire, 89 bytes captured)
Internet Protocol, Src Addr: 172.16.0.30 (172.16.0.30), Dst Addr: 172.16.16.8 (172.16.16.8)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 32817 (32817)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x1dedc30b
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
    Value: Timeticks: (463121126) 53 days, 14:26:51.26
```

SNMP
→ Protokollmittschnitt - GET NEXT: Übersicht

```
Frame 7 (85 bytes on wire, 85 bytes captured)
Internet Protocol, Src Addr: 172.16.16.8 (172.16.16.8), Dst Addr: 172.16.0.30 (172.16.0.30)
User Datagram Protocol, Src Port: 32817 (32817), Dst Port: snmp (161)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x1dedc30c
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
    Value: NULL
```

```
Frame 8 (124 bytes on wire, 124 bytes captured)
Internet Protocol, Src Addr: 172.16.0.30 (172.16.0.30), Dst Addr: 172.16.16.8 (172.16.16.8)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 32817 (32817)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x1dedc30c
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: STRING: netadmin@informatik.fh-gelsenkirchen.de
```

Inhalt

- Das Kommunikationsmodell von SNMP
- **Mängel und Probleme von SNMPv1**

Mängel und Probleme von SNMPv1

- Aufgrund des umfangreichen praktischen Einsatz von SNMP wurden sehr schnell die Mängel und Probleme des Netzmanagement deutlich.

- Die **wichtigsten Kritikpunkte von SNMPv1** sind:
  - Erhöhte Netzlast durch SNMP
  - Fehlende notwendige Sicherheit
  - Fehlende Funktionen in SNMP

Mängel und Probleme von SNMPv1
→ Erhöhte Netzlast durch SNMP

- Die für eine Beobachtung eines Netzes notwendigen Informationen erhält SNMP im wesentlichen durch **Polling**.
- SNMP sieht zwar die Möglichkeit von **Traps** vor, diese sind aber auf einige wenige Ereignisse im Agenten beschränkt.
- Die Entscheidung, Polling statt Traps zu benutzen, schafft für den praktischen Einsatz von SNMP vor allem bei großen Netzen Probleme, da bei großen Netzen sehr viele Polling-Requests nötig sind.
  - **Das Netz**, dessen Fehler eigentlich durch den Einsatz eines Netzmanagementsystems erkannt werden sollen, **wird belastet**, was wiederum der Grund für andere Probleme sein könnte.
- Ein weiterer Mangel bei SNMP ist die zentrale Ausrichtung der Managementstation.
  - Bei SNMP existiert unabhängig von der Größe des Netzes nur eine zentrale Netzmanagement-Station.
  - Damit muß eine einzige Station alle Knoten im Netz kontrollieren und überwachen.

Mängel und Probleme von SNMPv1
→ Fehlende notwendige Sicherheit

- Es gibt keine wirksamen Sicherheitsmechanismen, was zu den folgenden Gefahren führt:
  - **Maskerade**
    Vortäuschen der Identität, um Managementfunktionen durchführen zu können.
  - **Modifikation von SNMP-Paketen**
    Unerlaubtes Verändern von Parameterwerten in den SNMP-Paketen
  - **Veränderung der Reihenfolge**
    SNMP-Daten werden über den ungesicherten Dienst UDP transportiert.
  - **Spionage**
    Mitlesen von Managementdaten.
  - **Einfaches Passwortverfahren ohne Verschlüsselung**
    Ermöglicht eine Maskerade

Mängel und Probleme von SNMPv1
→ Fehlende Funktionen in SNMP

- SNMP kennt nur vier verschiedene Typen von Operationen: **get, get-next, set** und **trap.**
- Mit get und set können mehrere MOs durch einen atomaren Befehl abgerufen werden, aber keine komplexen Datenstrukturen (z.B. ganze Tabellen) oder Klassen von Daten.
- Dies führt in der Praxis des Netzmanagement, in der durchaus Datenkonstrukte vorkommen (etwa Routing-Tabellen), zu Einschränkungen.
- SNMP Operationen liefern zum Teil **wenig aussagekräftige** Fehlercodes.
- Traps werden in SNMP nicht bestätigt. Damit gibt es keine Rückkopplung, dass wichtige Traps ihr Ziel erreicht haben.

Netzwerkmanagement mit SNMP
→ Teil 2

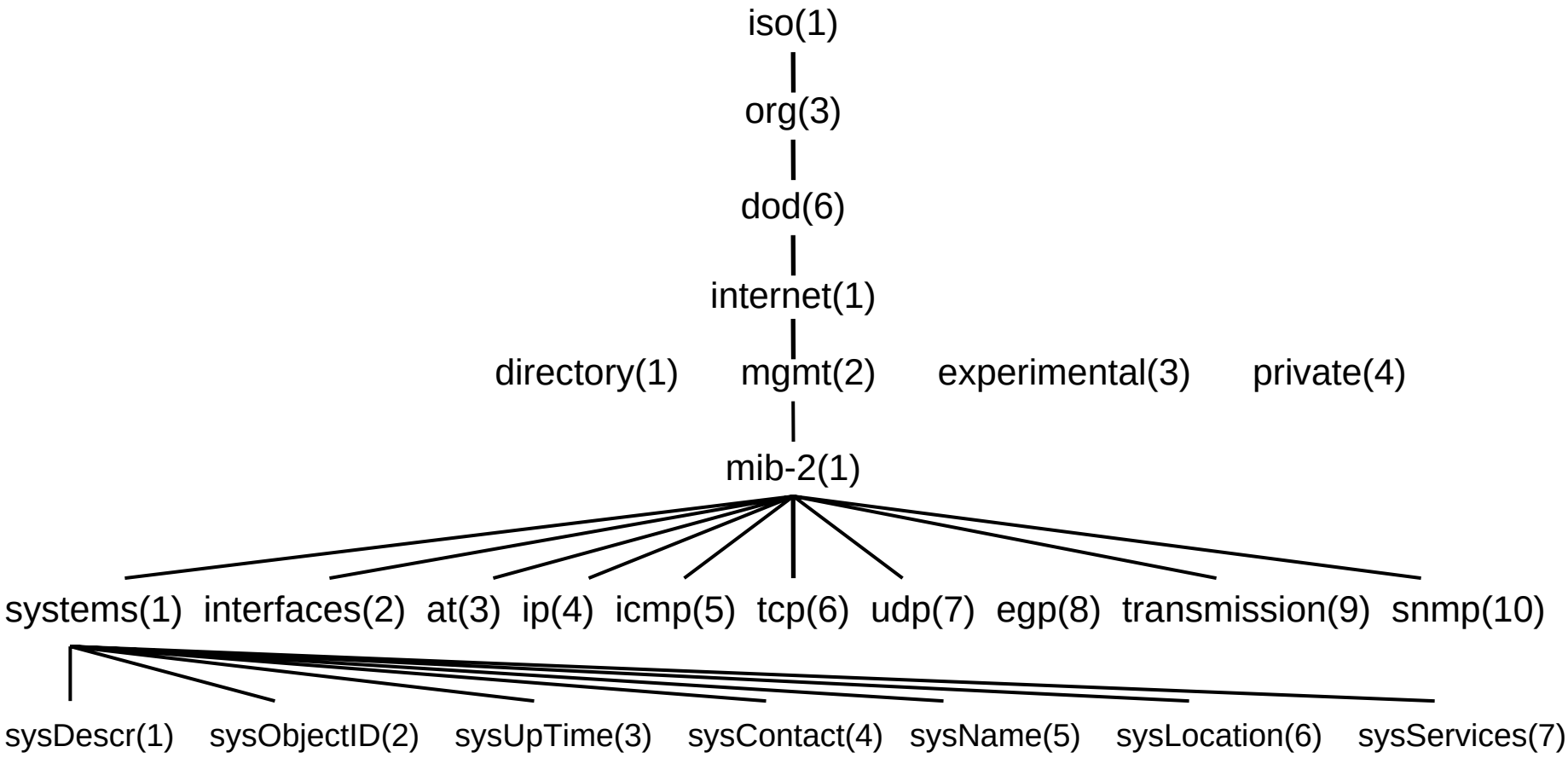
Vielen Dank für Ihre Aufmerksamkeit

Fragen ?

norbert.pohlmann@informatik.fh-gelsenkirchen.de

Fachhochschule Gelsenkirchen

Baum: systems



Baum: interface