# Netzmanagement

Einführung in das Management
verteilter Systeme

# Inhaltsverzeichnis



# 1 Management verteilter System – integriertes Management

## 1.1 Einleitung

***"The Network is the Computer"***

Dieser, zugegebenermaßen etwas provozierende Werbeslogan der Firma SUN Microsystems verdeutlicht, welche Bedeutung Computer-Netze mittlerweile erlangt haben. Obwohl der Werbeslogan eigentlich für Java als Programmiersprache für verteilte Anwendungen entworfen wurde, trifft die Aussage im Grunde die Bedeutung von vernetzten Systemen in vielen Bereichen: Ein hochverfügbares Rechnersystem nützt nichts, wenn die Dienste, die das System zur Verfügung stellt, aufgrund eines Netzproblems nicht genutzt werden können.

Die Realisierung einer Verwaltungsstruktur für bestehende Netze stellt oftmals ein Problem dar, da in vielen Unternehmen heute ein heterogenes Netz im Einsatz ist, das aus historischen Gegebenheiten entstanden ist. Typischerweise wurden im Laufe der Zeit zunächst an verschiedenen Standorten eines Unternehmens nach und nach diverse Rechner installiert. Abhängig von den speziellen Anforderungen der jeweiligen Abteilung wurden Rechnersysteme mit den passenden Applikationen ausgewählt. Steigender Kommunikationsbedarf und der Wunsch gemeinsame Ressourcen zu nutzen führt nun dazu, dass schrittweise immer mehr Rechner miteinander vernetzt werden. Dadurch entsteht ein unternehmensweites heterogenes Netz, das durch Verwendung von Netzkoppelelementen eine Ankopplung entfernter Unternehmensstandorte erlaubt. Das Netz wird zu einem zentralen und unverzichtbaren Element, dessen Störung und Ausfall nicht toleriert werden kann.

Da die Verfügbarkeit des Rechnernetzes für viele Unternehmen einen wichtigen Faktor darstellt und im Falle eines Fehlers die Arbeit im Unternehmen zum Erliegen kommen kann, ist man darauf bedacht, das Netz so gut wie möglich zu überwachen und zu steuern. Die Netzverantwortlichen benötigen dazu Instrumente, mit denen die **Zuverlässigkeit, Verfügbarkeit und Verwaltbarkeit eines Netzes erhöht** werden kann. Dies soll ein Managementsystem leisten, das möglichst alle Netzprobleme bewältigen und beliebige Netzkomponenten verwalten kann.

Ziele für den Netz-Betrieb sind:

- Gewährleistung der Netz-Verfügbarkeit
- Gewährleistung der Beherrschbarkeit der Netz-Technologien
- Optimierung der Durchlauf- und Bearbeitungszeiten
- Maximierung der Automatisierung

Ein Managementsystem sollte die Möglichkeit bieten, diese Ziele zu verfolgen und zu realisieren.

## 1.2 Integriertes Management

### 1.2.1 Definition des Begriffes integriertes Management

Natürlich stellt das Management des Netzes, wie im vorigen Abschnitt beschrieben, nur den ersten Schritt in Richtung des Managements der Informationsressourcen eines Unternehmens dar. Unter Management versteht man alle **Maßnahmen** für einen **effektiven und effizienten Betrieb eines verteilten Systems**. Integriertes Management bedeutet die Steuerung und Überwachung sämtlicher in einem verteilten System auftretender Ressourcen gemäß eines einheitlichen, durch eine integrierte Managementarchitektur vorgegebenen Vorgehens.

Der Begriff ‘integriert’ im Zusammenhang mit dem Management von Informationsverarbeitungs-Ressourcen betrifft dabei unterschiedliche Aspekte. So muss unter anderem der wichtige Aspekt der Herstellervielfalt (**Heterogenität**) bewältigt werden, weshalb ein integriertes Management zwangsläufig auch ein **offenes Management** sein sollte. Außerdem sollen durch ein integriertes Management sämtliche Funktionsbereiche, wie Konfigurations-, Fehler-, Abrechnungs- und Sicherheitsmanagement, aller Typen von Informationsverarbeitungs-Ressourcen (Netz-, System- und Anwendungs-Ressourcen) berücksichtigt werden.

Ziel eines integrierten Managements ist es, die unterschiedlichsten Managementobjekte und Disziplinen möglichst einheitlich zu behandeln.

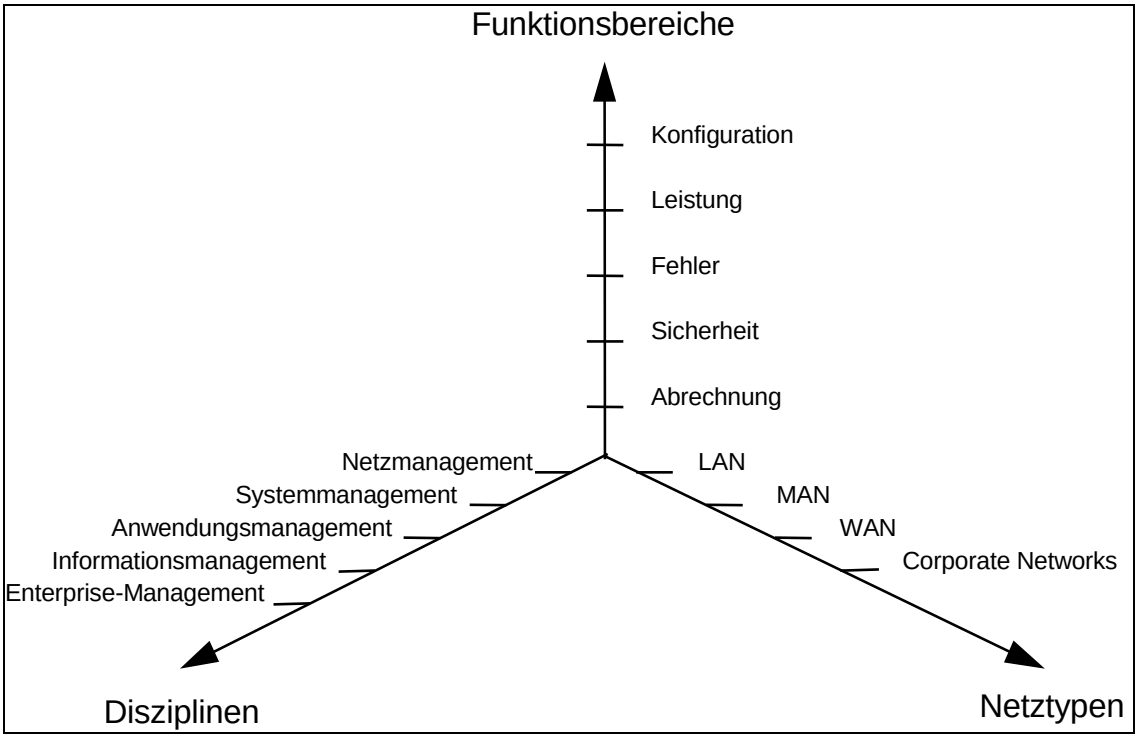



*Bild 1-1: Bereiche des technischen Managements*

## 1.2.2 Ebenen des integrierten Managements

Die Ebenen bzw. Disziplinen des integrierten Managements lassen sich in fünf Bereiche aufgliedern:

- Netzmanagement
- Systemmanagement
- Informationsmanagement
- Anwendungsmanagement
- Enterprise-Management

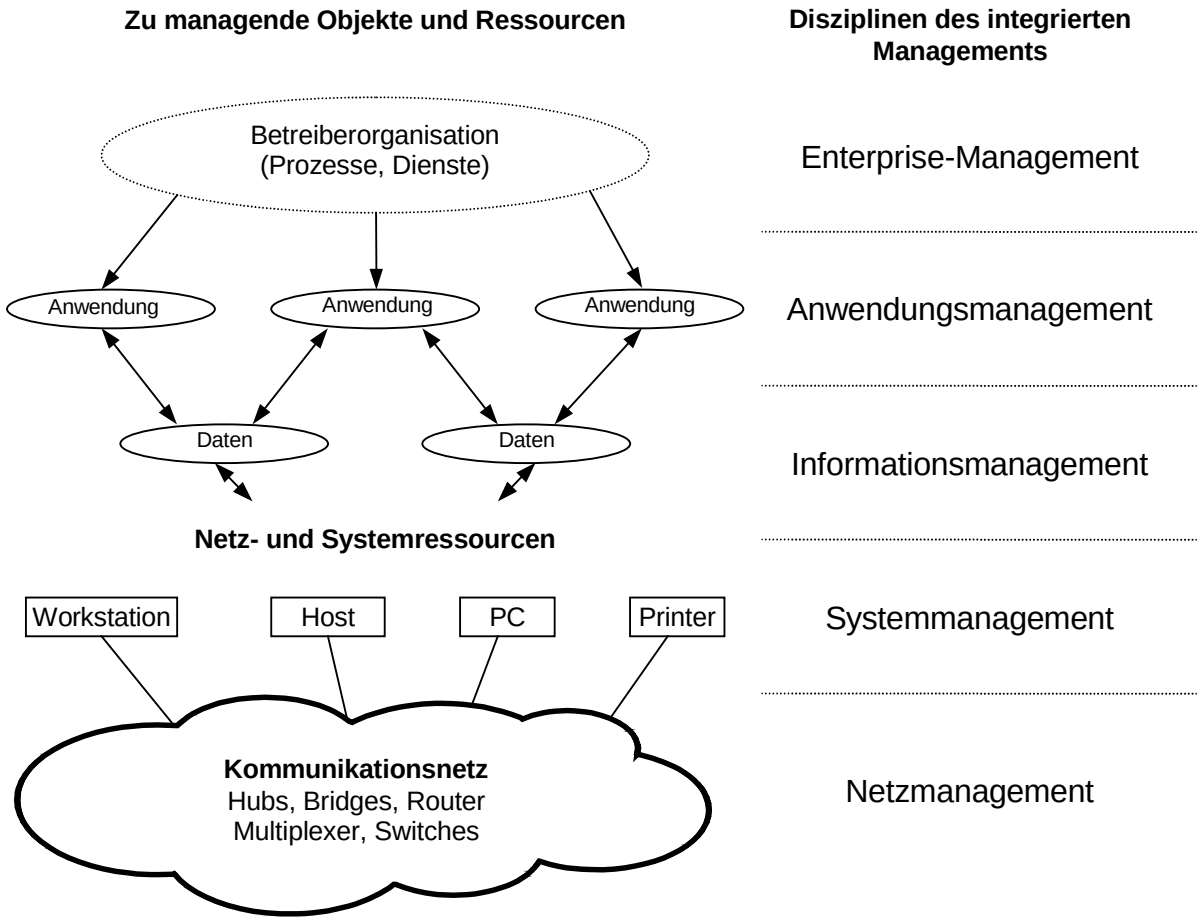



*Bild 1-2: Ebenen des integrierten Managements*

#### 1.2.2.1 Netzmanagement

Das Netzmanagement beschäftigt sich schwerpunktmäßig mit dem **Management von Kommunikationsdiensten und Netzkomponenten**. Es soll für einen möglichst reibungslosen Betrieb des Netzes sorgen und ist zuständig für Fehlersuche im Kommunikationsnetz, Konfigurationsverwaltung der Netzkomponenten und Planung des Kommunikationsnetzes. Zu verwaltende **Objekte** können z.B.

- Leitungen,
- Übertragungs- und Vermittlungseinrichtungen (Switches, Bridges, Router) und
- Protokollinstanzen

sein.

Man kann die **Funktionsbereiche des Netzmanagements** in fünf Gruppen unterteilen

- **Netzsteuerung** (operational management)

  beschreibt die Gruppe der Funktionen, die im laufenden Betrieb dazu benutzt werden, die Netzwerk-Betriebsmittel bereitzustellen und zu verwalten.

- **Fehlermanagement** (maintenance)

  fasst alle Funktionen zusammen, die zur Fehlervorbeugung, Fehlererkennung und Fehlerbehebung im Kommunikationsnetz verwendet werden können.

- **Konfigurationsverwaltung** (configuration management)

  gibt Hilfsmittel und Funktionen zur Planung, Erweiterung und Änderung der Konfiguration sowie zur Pflege von Konfigurations-Informationen an. Sie dient dazu, das gesamte vernetzte und verteilte Netz verfügbar zu machen.

- **Netztuning** (performance management)

  enthält Hilfsmittel und Werkzeuge zur Messung und Optimierung des Leistungsverhaltens des Kommunikationsnetzes.

- **Benutzerverwaltung** (user administration)

  beschreibt die Gruppe der Funktionen, die für die ordnungsgemäße Abwicklung der Benutzung des Kommunikationsnetzes, wie z.B. Zugangsverwaltung, Verbrauchskontrolle und Abrechnungshilfen, verantwortlich sind.

#### 1.2.2.2 System-Management

Das Systemmanagement befasst sich mit dem **Management der Ressourcen von Endsystemen** (z.B. Workstation, Host, PC, Printer) **und Systemverbunden.** Beispiele für die Aufgabengebiete des Systemmanagements sind

- Datenhaltung,
- Verteilung von Software,
- Lizenzkontrollen,
- Lastverteilung,
- Abrechnung und
- Alarmweiterleitung.

Man kann die **Funktionsbereiche des Systemmanagements** in ebenfalls fünf Gruppen unterteilen

- **Fehlererkennung**
  von Fehlern in Endgeräten, wie Server, Printer und Workstations.
- **Leistungsanalyse**
  zur Ermittlung potentieller und realer Engpässe in Endgeräten und Endsystemen.
- **Installation und entfernte Konfiguration**
  von Software von einer zentralen Stelle im Netzwerk.
- **Sicherheit und Überwachung von Endsystemen**
- **Inventur und Accounting**
  ermöglicht eine genaue und dynamische Inventur aller Endsysteme und Software am und im Netz.

**Objekte des Systemmanagements** sind z.B.

- CPUs,
- Speicher,
- Platten,
- Peripheriegeräte,
- Prozesse,
- Server,
- Benutzer und
- Dateisysteme.

#### 1.2.2.3 Anwendungsmanagement

Das Anwendungsmanagement ist zuständig für das **Management verteilter Anwendungen und verteilt realisierter Dienste**, wie z.B. Steuerungssysteme zur automatischen Fertigung, verteilt realisierte Informations- und Auskunftssysteme (Mailsysteme, Directories) oder Systeme zur Unterstützung von Gruppenarbeit.

Die Aufgabe des Anwendungsmanagements besteht darin, verteilte Anwendungen zu überwachen, zu aktivieren bzw. deaktivieren sowie eine Umverteilung von Teilen einer Anwendung zur Laufzeit durchzuführen. Dies kann durch das Starten, Beenden oder Abbrechen von Prozessen, Prozessgruppen oder der Gesamtanwendung veranlasst werden.

Trotz seiner großen Bedeutung ist das gesamte Gebiet des Anwendungsmanagements größtenteils noch Gegenstand der Forschung.

#### 1.2.2.4 Integration des Anwendungsmanagements in Netz- und Systemmanagementlösungen

Dem Ansatz zur Integration eines Anwendungsmanagements in eine bestehende Netz- und Systemmanagementlösung liegt eine einfache Idee zugrunde:

Die Prozesse einer verteilten Anwendung werden wie sämtliche andere Netz- und Systemressourcen als Managementobjekte angesehen und können auch in gleicher Weise von einer Manageranwendung verwaltet werden.

Die begriffliche Grenze zwischen Systemmanagement und Anwendungsmanagement ist unscharf, wenn der verteilt realisierte Dienst ein allgemeiner „Systemdienst“ ist.

#### 1.2.2.5 Enterprise-Management

Unter dem Enterprise-Management ist das **Management eines unternehmensweiten Kommunikationsnetzes** (Corporate Network) zu verstehen. Man sieht beim Enterprise-Management nicht nur den technischen Aspekt der Zusammenführung unterschiedlicher Managementsysteme, sondern verknüpft die Managementproblematik mit dem gesamten Unternehmenskontext, in dem nicht nur technische, sondern auch organisatorische und betriebs- und volkswirtschaftliche Faktoren mit in Betracht zu ziehen sind.

Letztlich ist nicht das Kommunikationsnetz oder dessen Komponenten der Kern des Managements, sondern die Geschäftsvorgänge, die durch das Zusammenspiel der Komponenten unterstützt werden sollen. Nur durch ein übergreifendes Management, das alle Teilbereiche überwacht und steuert, kann die Qualität und Verfügbarkeit eines Geschäftsprozesses gewährleistet werden. Probleme ergeben sich hier vor allem an den Schnittstellen der einzelnen Managementbereiche, da diese von den einzelnen Herstellern nicht für die Kommunikation untereinander ausgelegt wurden.

Momentan existieren auf dem Gebiet des Enterprise-Managements mehr ungelöste Fragen als Antworten, insbesondere gibt es bis heute noch kein Managementsystem, welches die Probleme praktisch lösen könnte. Nur wenige Hersteller, wie etwa Hewlett Packard oder Tivoli, haben überhaupt die Erfahrung und den finanziellen Hintergrund, um auf diesem Gebiet eine Lösung entwickeln zu können.

## 1.3 Funktionsmodelle

Das Funktionsmodell einer Managementarchitektur zergliedert den Gesamtaufgabenkomplex Management in **Management-Funktionsbereiche**.

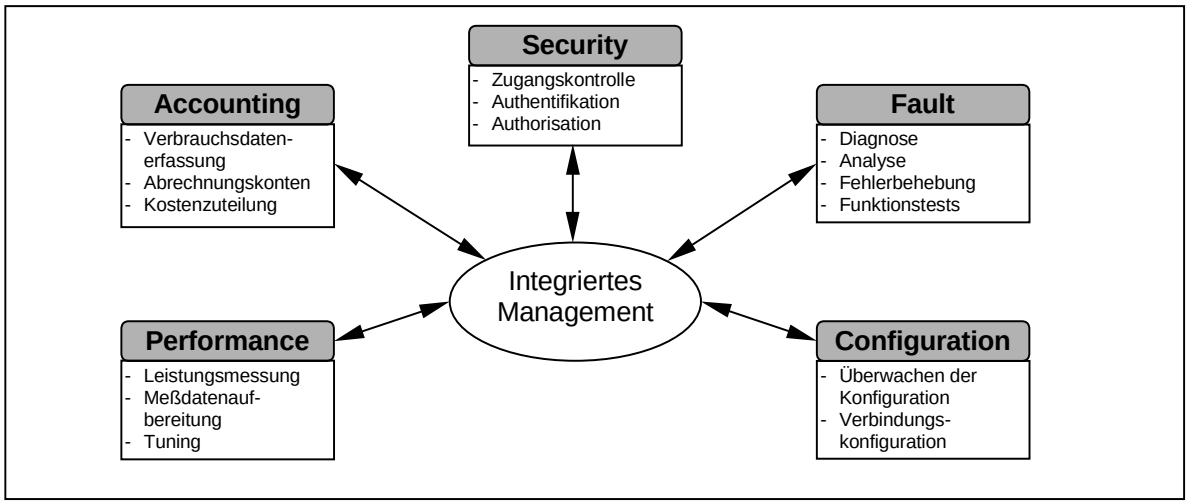



*Bild 1-3: Funktionsbereiche des integrierten Managements*

### 1.3.1 Performance Management

Die Aufgabe des Performance Managements (Leistungsmanagement) ist die **Analyse und Konfiguration der Leistung eines Rechners oder** eines **Netzwerkes**. Im einzelnen umfasst dies Funktionen zur

- Messung von Performance-Daten,
- Aufbereitung von Performance-Daten,
- Analyse und Tuning von Netzkomponenten,
- Erkennung von Engstellen und Leistungslöchern des Systems und
- Aufstellung von Performance-Prognosen bei geplanter Änderung des Netzes.

### 1.3.2 Fault Management

Das Fault Management (Fehlermanagement) beschäftigt sich mit der **Fehlererkennung** und wenn möglich auch mit der **Fehlerbehebung**. Hierunter fallen Funktionen zur

- Problemerkennung,
- Durchführung von Netzwerk- und Komponententests,
- Problembehebung und
- Überwachung von Netzwerk und Komponenten (Monitoring).

### 1.3.3 Configuration Management

Aufgabe des Configuration Managements ist die **Kontrolle der Konfiguration** des Netzwerkes sowie der daran angeschlossenen Komponenten. Es umfasst Funktionen für

- Lastkonfiguration und Überwachung der Konfiguration
- Verbindungskonfiguration (Routing)
- Automatische und ferngesteuerte (remote) Konfiguration

### 1.3.4 Accounting Management

Handelt es sich bei dem zu verwaltenden Netz um ein firmen- oder filialen-übergreifendes Netz, so benötigt man ein Accounting Management (Gebühren-management). Das Accounting Management beinhaltet Funktionen zur **Gebühren-erfassung**, der Zuordnung der Gebühren zu den jeweiligen Nutzern und der Verteilung zur Rechnungserstellung.

### 1.3.5 Security Management

Ein oft sträflich vernachlässigter Bereich ist das Security Management (Sicherheits-management). Aufgabe des Sicherheitsmanagements ist, alle Dienste des Netzwerkes und deren Komponenten hinsichtlich sicherheitsrelevanter Kriterien zu konfigurieren. Dazu zählen Funktionen zu folgenden Gebieten

- Zugangskontrolle zu bestimmten Diensten
- Authentifikation (Erkennung der Benutzer)
- Authorisation (Vergabe von Rechten an erkannte Benutzer)
- Passwörter

## 1.4 Managementarchitekturen und ihre Teilmodelle

### 1.4.1 Allgemeines

Ein Rahmenwerk für managementrelevante Standards wird **Management-architektur** genannt.

Managementarchitekturen sind Voraussetzung für den Entwurf von Management-systemen in heterogener Umgebung. Verteilte Systemumgebungen unterscheiden sich extrem, was ihren Aufbau, ihre Größe oder ihre Ausrichtung betrifft. Es kann folglich auch **nicht die eine Managementlösung für alle verteilten Systeme** geben. Um für jede Umgebung zu einer optimalen Managementlösung zu kommen, muss ein Baukasten von Modulen entworfen werden, der so flexibel wie möglich zusammengesetzt werden kann. Bezüglich diesen Hintergrundes ist es klar, dass es zu unterschiedlichen Ausprägungen von Managementarchitekturen kommen kann.

Alle Architekturen müssen aber grundsätzlich zu folgenden Aspekten bzw. Modellen Festlegungen treffen:

- Organisationsmodell
- Informationsmodell
- Kommunikationsmodell
- Funktionsmodell

## 1.4.2 Organisationsmodell

Das Organisationsmodell einer Managementarchitektur **legt die Akteure, ihr Rollenspiel und die Grundprinzipien ihrer Kooperation fest**

### 1.4.2.1 Kooperationsformen

Derzeit findet man für das Management verteilter heterogener Systeme zwei unterschiedliche Kooperationsformen.

- **Manager-Agenten-Modell**
  Das Manager-Agenten-Modell unterstellt mit seiner **symmetrischen/ hierarchischen** Kooperationsform ein **Auftraggeber-Auftragnehmer-Verhältnis** für eine Kooperationsbeziehung. Der Manager beauftragt den Agenten, eine bestimmte Operation auszuführen oder Informationen bereitzustellen. Der Agent antwortet dann mit dem Ergebnis der Operation oder der gewünschten Information. Die Rollen sind den Akteuren häufig nicht statisch zugeordnet, sondern können sich dynamisch je nach Aufgabe ändern. Diese Kooperationsform ist derzeit im Bereich des Managements am häufigsten vertreten. Beispiele sind das **OSI-Management (CMIP)** und das **Internet-Management (SNMP)**.
- **Peer-to-Peer-Modell**
  Beim Peer-to-Peer-Modell wird eine **symmetrische Kooperationsform** angewandt, die von einer Kommunikation und Kooperation prinzipiell **gleichberechtigter Objekte** ausgeht. Man findet hier eine flexible, wechselseitige Auftragsbeziehung mit Informationsaustausch in beide Richtungen.

### 1.4.2.2 Domänenkonzept

Da **unterschiedliche Management-Sichten auf Ressourcen** oftmals vorteilhaft für das Managen von Systemen sind, besteht die Möglichkeit organisatorisch gleichartige Ressourcen der zu verwaltenden Objekte in sogenannte **Domänen** zu gruppieren.

Falls ein Organisationsmodell ein Domänenkonzept vorsieht, muss es festlegen, wie solche Gruppen gebildet werden, wie den Domänen Zuständigkeiten zugeordnet werden und wie Zuordnungen dynamisch modifiziert werden können.

### 1.4.3 Kommunikationsmodell

Das Kommunikationsmodell muss die **kommunizierenden Partner** und die **Kommunikationsmechanismen** für folgende Kommunikationszwecke festlegen:

- Abfragen von Informationen von Ressourcen,
- Austausch von Steuerinformationen und
- asynchrone Ereignismeldungen.

Folglich müssen also **Dienste und Protokolle** für Management-Anwendungen spezifiziert werden. Ferner müssen die Syntax und die Semantik der Protokolldateneinheiten (PDUs) definiert und die Managementprotokolle in eine Protokollhierarchie einer zugrundeliegenden Kommunikationsarchitektur ( z.B. TCP/IP, OSI) eingebettet werden.

### 1.4.4 Informationsmodell

Das **Informationsmodell spezifiziert eine Modellierungs- und Beschreibungsrahmen für zu verwaltende Objekte** (Managementobjekte). Aus Managementsicht müssen nicht der gesamte Aufbau oder alle internen Abläufe von Ressourcen bekannt sein. Es müssen nur managementrelevante Parameter, wie z.B. Konfigurations- und Tuningparameter und Funktionen, wie „Start“, „Stop“ und „Aktiviere Datensicherung“ modelliert werden.

Die Managementsicht kann somit als ein Modell (Abstraktion) der realen HW/SW-Ressourcen für Managementzwecke aufgefasst werden.

Für ein Managementobjekt wird im Informationsmodell festgelegt

- wie es identifiziert werden kann,
- wie es zusammengesetzt ist,
- wie es sich verhält,
- wie es manipuliert werden kann,
- welche Beziehungen zu anderen Managementobjekten bestehen und
- wie es über das Managementprotokoll angesprochen werden kann.

Das Informationsmodell einer Managementarchitektur legt folglich den **Modellierungsansatz** und eine **eindeutige Syntax für die Beschreibung von Managementinformationen** fest.

In der **Managementinformationsbasis** (MIB) werden die von einem Manager- bzw. Agentensystem **verwaltete Menge von Managementobjekten** beschrieben. Man versteht unter der MIB eine virtuelle Datenbank, die konkrete Beschreibungen für Managementobjekte enthält. Die einzelnen Objekte sind in Gruppen angeordnet.

## 1.4.5 Funktionsmodell

Das Funktionsmodell einer Managementarchitektur zergliedert den Gesamtaufgabenkomplex Management in Management-Funktionsbereiche (z.B. Konfiguration-, Fehler-, Abrechnungs- und Leistungs-Management) und versucht, allgemeine Managementfunktionen bereichsspezifisch festzulegen.

Im Funktionsmodell sind für die einzelnen **Funktionsbereiche** die erwartete Funktionalität und die Dienste, sowie Managementobjekte zur Erbringung der Funktionalität festzulegen. Des weiteren ist die **Aufrufkonvention für Managementfunktionen** zu spezifizieren.

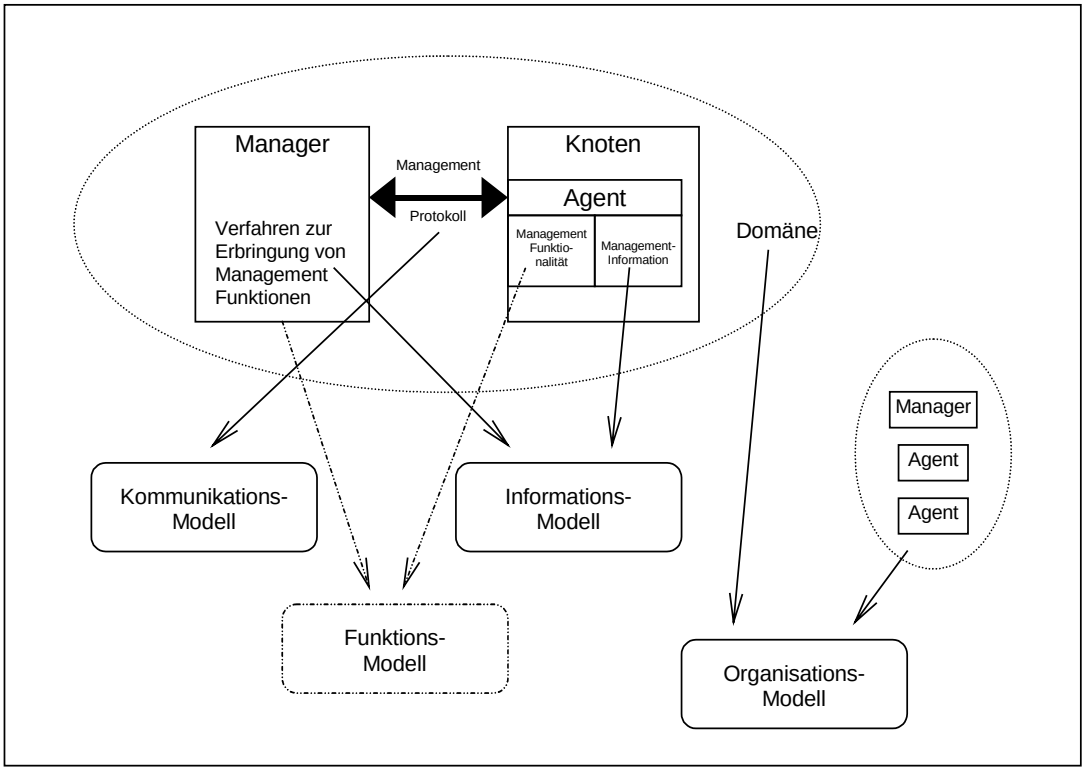



*Bild 1-4: Teilmodell einer integrierten Managementarchitektur*

## 1.4.6 Existierende Managementarchitekturen

Die wichtigsten herstellerunabhängigen Managementarchitekturen sind folgende

- **OSI-Management**

  das OSI-Management ermöglicht es unterschiedlichste Netztypen (z.B. LAN, MAN, WAN) integriert zu verwalten.

- **Internet-Management** der IAB/IETF

  hat sich allgemein als alleiniger Management-Vertreter lokaler Netze (LAN) durchgesetzt

- **TMN** (Telecommunications Management Network) der ITU

  dient zum Verwalten von öffentlichen Telekommunikationssystemen.

- **OMA/CORBA** (Object Management Architecture) der OMG

  diese Managementarchitektur wurde für allgemeine objektorientierte verteilte Anwendungen konzipiert und erlaubt in erster Linie nicht ortsgebundene Kooperation von Objekten in verteilter Umgebung

- **DMI** (Desktop Management Interface) der DTMF

  wurde entwickelt, um die aus der Sicht des standardisierten Managements nicht verwaltbaren PCs verwalten zu können. Ziel des DMI ist es, heterogene PCs und Workstations in ein integriertes Management einzubinden.

# 2 Das OSI-Referenzmodell

Das OSI-Referenzmodell, auch als ISO/OSI-Referenzmodell oder OSI Basic Reference Model (OSIBRM) bezeichnet, ist das Kommunikationsmodell der International Standard Organisation (ISO). Es modelliert das Kommunikationssystem eines Rechners und seine Zusammenarbeit mit dem Kommunikationssystem eines anderen Rechners. Das **Kommunikationssystem (communication system)** eines Rechners ist für den rechnerübergreifenden Austausch von Nachrichten verantwortlich und nutzt zur Übertragung ein Rechnernetz.

Ein Rechnernetz, im folgenden auch kurz Netz genannt, wird durch die folgenden Anforderungen charakterisiert:

- Ein Rechnernetz ist primär ein **Transport-** und **Übertragungssystem** für den Austausch digitaler Daten zwischen an das Netz angeschlossenen, weitgehend oder vollständig autonomen Teilnehmern.
- Ein Rechnernetz bietet seinen Teilnehmern die Möglichkeit, nacheinander oder auch gleichzeitig mit jedem anderen gewünschten Netzteilnehmer zum Zweck des Datenaustausches in Verbindung zu treten.
- Realisiert wird ein Rechnernetz in der Regel durch eine Anzahl von **Netzknoten**, d.h. speziell für die Aufgaben der Vermittlung und Übertragung digitaler Daten entwickelten DV-Systeme und den **Verbindungen** der Knoten untereinander.

## 2.1 Offene Systeme nach ISO/OSI

Bereits in den 60er Jahren waren teilweise weltweite Rechnernetzwerke vorhanden. Als Beispiel seien etwa das bekannte ARPA-Netz (Forschungsnetz), verschiedene Herstellernetze oder Rechnerverbunde von Fluglinien und Banken erwähnt. Alle Herstellernetze hatten eines gemeinsam: Sie erfüllten ihre Aufgabe nur für einen sehr beschränkten Teilnehmerkreis. Es war Teilnehmern verschiedener Netzwerke nicht möglich, miteinander zu kommunizieren. Jedes dieser Netze samt ihren Datenendgeräten bildete ein sogenanntes geschlossenes System.

Ein **geschlossenes System** hat folgende Merkmale: es baut auf der Technologie eines Herstellers auf und ist mit Konkurrenzprodukten nicht kompatibel. Die Ausdehnung eines derartigen Netzes beschränkt sich auf einen bestimmten Teilnehmerkreis und oft auf ein beschränktes räumliches Gebiet. Datenübertragung von einem Netz in ein anderes ist nur mittels verhältnismäßig großem Aufwand an Hard- und Software möglich.

Im Jahre 1977 erkannte die **ISO (International Standard Organization)** die Notwendigkeit, den Bereich der Rechnerkommunikation zu normen. Im "Subcommittee 16" , **Open Systems Interconnection (OSI)**, wurde die Arbeit aufgenommen. Dabei war zunächst das Ziel, die Kommunikation von Endsystemen verschiedener Hersteller, die über ein Fernmeldenetz (Wide Area Network) miteinander kommunizieren, zu normieren, damit Rechner verschiedener Hersteller in der Lage sind, miteinander Informationen über das Netz auszutauschen.

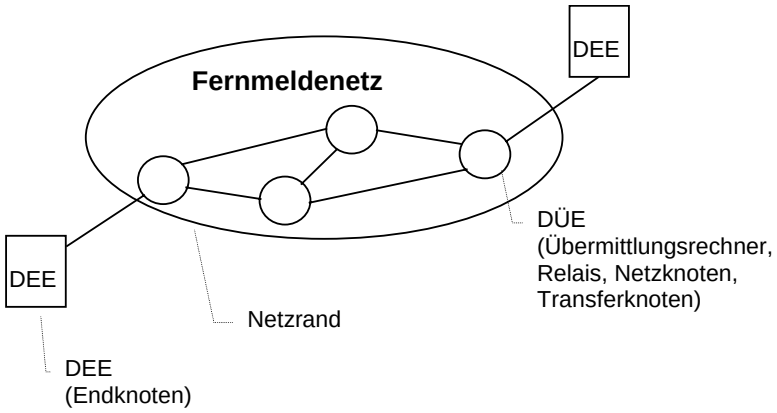



*Bild 2-1 Rechner, die Datenendeinrichtungen (DEE) am Netzwerkrand des Fernmeldenetzes sind*

Durch die Standardisierung sollte das Außenverhalten eines Systems, d.h. sein Kommunikationsverhalten, standardisiert werden. Mit anderen Worten, jeder Rechner soll sich nach außen gleichartig verhalten, ganz egal, von welchem Hersteller er kommt. Durch diese Standards soll erreicht werden, dass Systeme verschiedener Hersteller problemlos miteinander kommunizieren können.

Ein **offenes System** nach ISO/OSI ist also ein System, welches trotz herstellerabhängiger Implementierung mit anderen offenen Systemen nach festgelegten Normen kommuniziert.

Das Außenverhalten eines Rechners zeigt sich darin, dass er auf gewisse Reize (Telegramme) von außen, ein bestimmtes Verhalten zeigt, d.h. auf bestimmte Weise antwortet. Rein theoretisch hätte sich OSI darauf beschränken können, nur das **Außenverhalten** zu standardisieren (**Black-Box-Sicht**).

OSI ist etwas weiter gegangen. Wenn man nach außen passend reagieren soll, so muss man natürlich auch im Inneren des Rechners gewisse Arbeitseinheiten (Instanzen) haben, die für das Außenverhalten verantwortlich sind. OSI baute ein logisches Modell von **Arbeitseinheiten (Instanzen) eines Rechners** auf und sagte, welche Arbeitseinheit nach außen was tut. Jeder Rechner, mit dem man redet und der dem OSI-Modell genügt, soll dieselben Instanzen haben.

Dies bedeutet, dass OSI eine **White-Box-Sicht** für die Logik, nicht für die Implementierung des Kommunikationssystems hat. Man sieht in das Kommunikationssystem hinein und sieht die Instanzen:

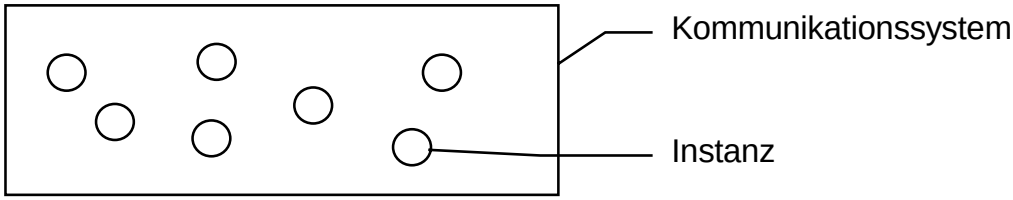



*Bild 2-2 Instanzen als Arbeitseinheiten eines Kommunikationssystems*

Da OSI sich für die White-Box-Sicht entschied, bedeutete dies mehr Arbeit als eine Black-Box-Sicht. Die Logik im Inneren sollte ja klar strukturiert und auch überzeugend sein. Strukturierung bedeutet immer Zerlegung und Festlegung des Zusammenwirkens der Zerlegungsprodukte. Die Problemstellung der Modularisierung des Kommunikationssystems führte dann zur Zerlegung des Kommunikationssystems in 7 Schichten und damit auch zur Festlegung des Zusammenwirkens der 7 Schichten (Schichtenmodell nach ISO/OSI).

Erläuterung:

**BOX**

Bei einer Black-Box sieht man nur die Box selbst, nicht jedoch, was sich in ihrem Inneren verbirgt. Bei einer White-Box schaut man in das Innere der Box.

- ISO/OSI regelt das Außenverhalten des Kommunikationssystems.
- ISO/OSI zerlegt das Kommunikationssystem in Schichten.
- ISO/OSI legt das Zusammenwirken der Schichten fest.
- ISO/OSI muss für das Außenverhalten nicht nur die Logik, sondern auch den konkreten Aufbau einer Nachricht festlegen.
- ISO/OSI muss sich im Innern des Kommunikationssystems auf die Logik beschränken, da Kommunikationssystem und Betriebssystem oftmals herstellerspezifisch eng zusammenarbeiten.

Schon recht bald stellte sich heraus, dass die Modellierung einer verbindungsorientierten Kommunikation wie z.B. beim Telefon, nur ein Ausschnitt aus der Realität war. Auch eine verbindungslose Kommunikation, sei es in einem Funknetz oder in einem Lokalen Netz (LAN), musste im ISO/OSI-Modell berücksichtigt werden. Ganz generell standardisiert das ISO/OSI-Modell also die Kommunikation in heterogenen Netzen. Heterogene Netze sind dabei Netze, die Stationen verschiedener Hersteller enthalten.

Die Modellierung jeglicher Kommunikation in heterogenen Netzen hatte zur Konsequenz, dass schon recht bald zum einen Subschichten (horizontale Strukturen), zum anderen aber alternative Möglichkeiten innerhalb einer Schicht

(vertikale Strukturen) je nach Art der Kommunikation berücksichtigt werden mussten. Je nach Kommunikationsart kommen dann die spezifischen alternativen Möglichkeiten in einer Schicht zum Einsatz. Während also das berühmte 7-Schichtenmodell horizontale Schnitte durch das Kommunikationssystem beschreibt, beschreibt ein **Protokollstack**, welche Protokolle und welche Protokollteilmengen in welcher Schicht zum Zuge kommen und mit welchen zusätzlichen Vereinbarungen mögliche Varianten in einer Schicht ausgegrenzt werden. Dies ist für eine eindeutige Lösung erforderlich, da die Standards für die Protokolle in den einzelnen Schichten meist Optionen und Wahlmöglichkeiten beinhalten. Durch diese Festlegungen wird ein Protokollstack aufgebaut mit dem Ziel, eine bestimmte Kommunikationsfunktion zu erhalten. Solche Festlegungen werden auch als **funktionaler Standard** oder **Profil** bezeichnet. So gibt es beispielsweise in der Schicht 7 für die Dienste:

- File Transfer, Access und Management
- Message Handling
- Virtual Terminal
- Directory
- etc.

spezielle Profile.

Ein funktionaler Standard (ein Profil) beschreibt also einen Protokollstack, dabei werden in jeder Schicht des Protokollstacks bestimmte einzelne Profile festgelegt. Beispiele funktionaler Standards sind MAP (Manufacturing Automation Protocol) und TOP (Technical and Office Protocols). Sie gehen auf die Initiative großer Anwender zurück, nämlich auf General Motors bei MAP und Boeing bei TOP.

## 2.2 Das ISO/OSI Basic Reference Model

Das **ISO/OSI Basic Reference Model**, auch bekannt unter dem kürzeren Namen **ISO/OSI Reference Model** darf die Implementierung des Kommunikationssystems nicht festlegen, da jeder Rechner seine eigene Architektur hat. Was es jedoch festlegen kann, ist ein allgemeines, abstraktes Modell, welches zum erwünschten Verhalten nach außen gemäß der gewünschten Verhaltensregeln führt. Dabei müssen die Verhaltensregeln vollständig angegeben sein, um ein definiertes Verhalten zu erreichen.

Die Frage der Modularisierung der Funktionalität eines Kommunikationssystems wurde nach einem „aristokratischen Prinzip“ durchgeführt. Es werden horizontale Schichten geschaffen. **Eine Instanz einer höheren Schicht hat als „Diener“ die Schicht darunter**. Dies bedeutet, dass sie Aufträge an ihren „Diener“ gibt und von ihm wieder Ergebnisse erhält.

Das Gesagte kann man auch etwas abstrakter formulieren:

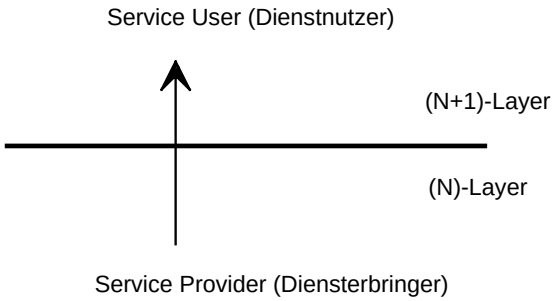



*Bild 2-3 Services und Schichten*

Es ist nicht begründbar, warum das ISO/OSI-Modell gerade 7 Schichten hat. Die Architekten der TCP/IP-Architektur erfanden 4 Schichten, die von DECnet erfanden 8 Schichten, SNA aus der IBM-Welt hat wiederum andere Schichten. Die Zahl der Schichten von ISO/OSI ist letztendlich auch nicht relevant, entscheidend ist aber, dass die verschiedenen Hersteller alle dieselben Schichten gemäß ISO/OSI implementieren, um damit interoperabel mit Rechnern anderer Hersteller zu werden.

Im folgenden seien diese Schichten einfach dargestellt:

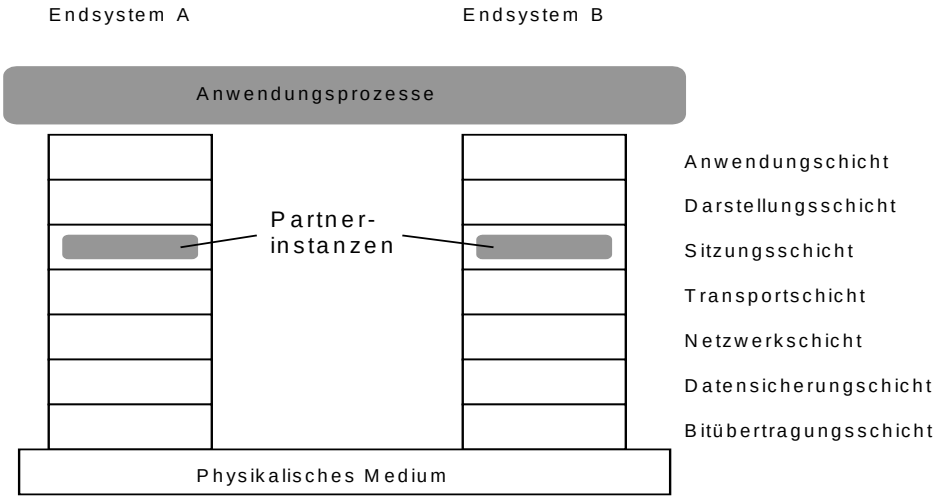



*Bild 2-4 Prinzipieller Aufbau des OSI-Schichtenmodells*

Es sei an dieser Stelle aber bereits darauf hingewiesen, dass man im Schichtenmodell zum einen „aristokratische Prinzipien“ (siehe Diener oder Services) entdecken kann, dass man aber auch „streng demokratische Prinzipien“ entdecken kann. Während innerhalb eines Rechners gedient wird, verhalten sich ganze Rechner am Netzwerkrand gegeneinander als gleichberechtigte unabhängige Wesen. Eine Instanz in einer Schicht des einen Rechners redet mit der entsprechenden Instanz in derselben Schicht des anderen Rechners als gleichberechtigtem Partner (**peer-to-peer communication**).

Nicht jedes Schichtenmodell ist gleich auszulegen. Daher einige Worte zur Interpretation:

- In jeder Schicht gibt es Instanzen (Arbeitseinheiten, entities). Diese Instanzen erbringen die schichtspezifischen Leistungen.
- Die Schichtung ist streng hierarchisch. Ein Instanz der Schicht N (N-Instanz) kann nur eine Instanz der Schicht N-1 darunter als Diener haben. Mit anderen Worten, eine N-Instanz kann nur den Dienst einer (N-1)-Instanz in Anspruch nehmen.
- Genauso kann eine N-Instanz ihre Dienste nur einer Instanz der direkt darüber liegenden Schicht anbieten, also einer (N+1)-Instanz.

## 2.3 Struktur des OSI-Modelles

[ Quelle : Helmut Kerner - Rechnernetze nach OSI ]

### 2.3.1 Instanzen

OSI zerlegt das Kommunikationssystem in 7 Schichten. In der OSI-Terminologie treten Stationen, Rechner, Prozesse oder Benutzer nicht auf, sondern man spricht von **entities (Instanzen)**.

In jeder Schicht eines jeden offenen Teilsystems existieren **Arbeitseinheiten (Instanzen, entities)**, die sowohl (vertikal) mit darüber- und darunterliegenden Instanzen als auch (horizontal) mit räumlich getrennten Instanzen derselben Schicht verkehrt. Diese **Instanzen (entities)** stellen logische parallele Prozesse in einer Schicht dar, welche die Aufgaben der jeweiligen Schicht durchführen können. Eine Schicht eines Endsystems kann mehrere Instanzen enthalten. Die Implementierung einer Instanz kann in Hard- oder Software erfolgen.

Zwei **vertikal benachbarte Instanzen nennt man obere und untere Folgeinstanz**. Korrespondierende Instanzen derselben Schicht in verschiedenen Systemen werden **Partnerinstanzen (peer entities)** genannt.

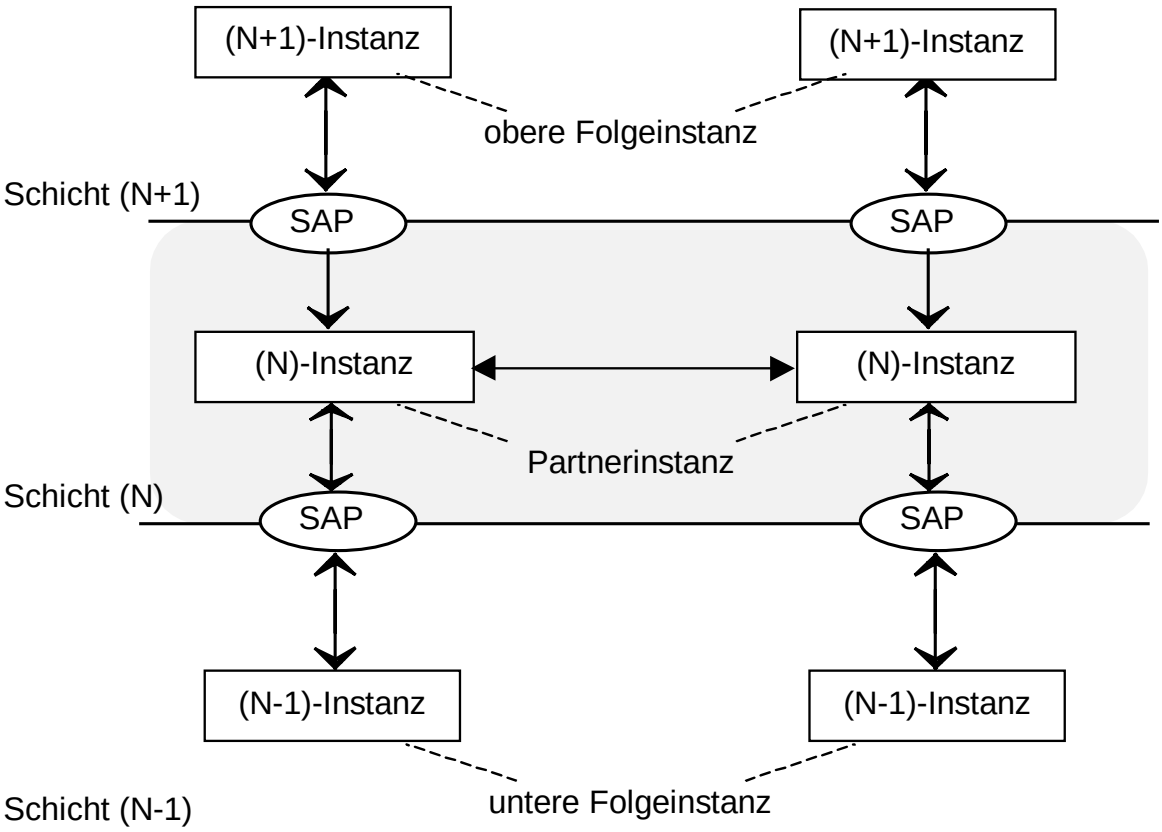



*Bild 2-5 Instanzen im OSI-Modell*

Aus dieser Struktur ergeben sich zwei Beziehungen zwischen Instanzen:

- **eine vertikale Beziehung, Dienst** genannt,
- und eine **horizontale Beziehung,** als **Protokoll** bezeichnet.

Jede Schicht N oder exakter jede Instanz in einer Schicht N bietet der übergeordneten Schicht N+1 **Services (Dienste)** an. Die übergeordnete Schicht N+1 ist der Nutzer der Dienste, die an der nächst tieferen Schicht N an den **Service Access Points (Dienstzugangspunkten)** zur Verfügung gestellt werden und durch alle Schichten, die unter der nutzenden Schicht liegen, erbracht werden. Die aufrufende Schicht - der Nutzer - heißt **Service User (Dienstnutzer)** und die genutzte Schicht heißt **Service Provider (Diensterbringer).**

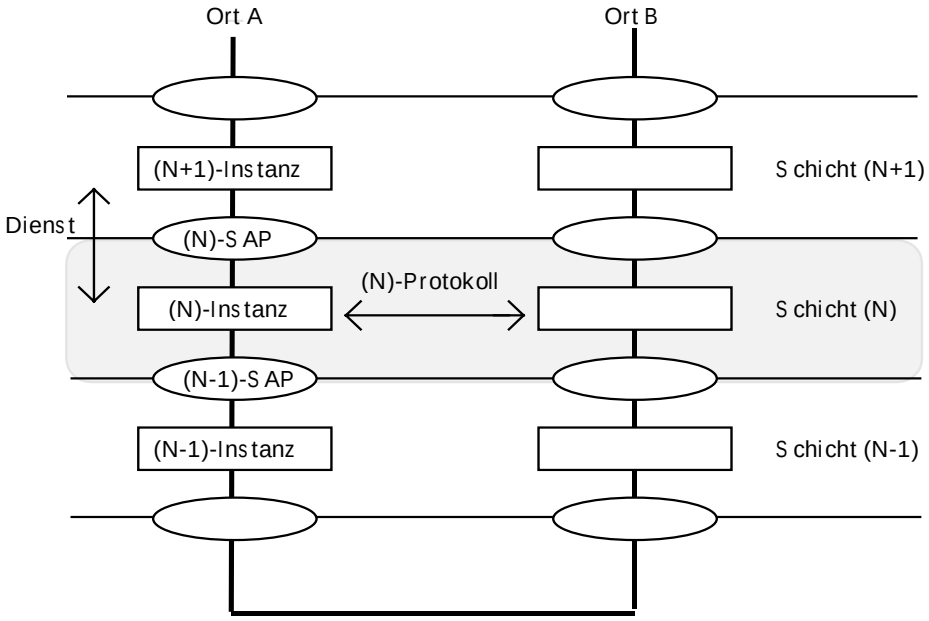



*Bild 2-6 Dienstzugangspunkte (Service Access Points), Instanzen und N-Schicht-Protokolle*

## 2.3.2 Dienste

[Quelle : T. Rose - Verwaltung von TCP/IP-Netzen]

Eine Schicht N liefert der darüber liegenden Schicht N+1 eine eigene Leistung, sowie die Leistungen aller darunter liegenden Schichten. Diese Leistung nennt OSI den **Dienst (service)** der Schicht. Es besteht z.B. die Leistung der Netzwerkschicht in der Vermittlung zwischen mehreren Orten und (mittels der Leistung der darunter liegenden Schicht) in der Erkennung und Korrektur von Fehlern in der Datenübertragung. **Die Kommandos zur Inanspruchnahme von Diensten und die Ergebnismeldungen heißen Dienstelemente (service primitives)**. Ein Dienst stellt hierbei einen Satz von Funktionen dar, die einem Benutzer von einem Anbieter (Provider) zur Verfügung gestellt werden.

Die Eigenschaften eines Dienstes werden durch die Schnittstelle am **SAP (service access point)** definiert. Eine N-Schicht nimmt also die Dienste einer (N-1)-Schicht in Anspruch, ohne zu wissen, wie diese ihre Dienstleistungen erbringt (Prinzip des Information Hiding). Genauso stellt eine N-Schicht der (N+1)-Schicht ihre Dienste zur Verfügung.

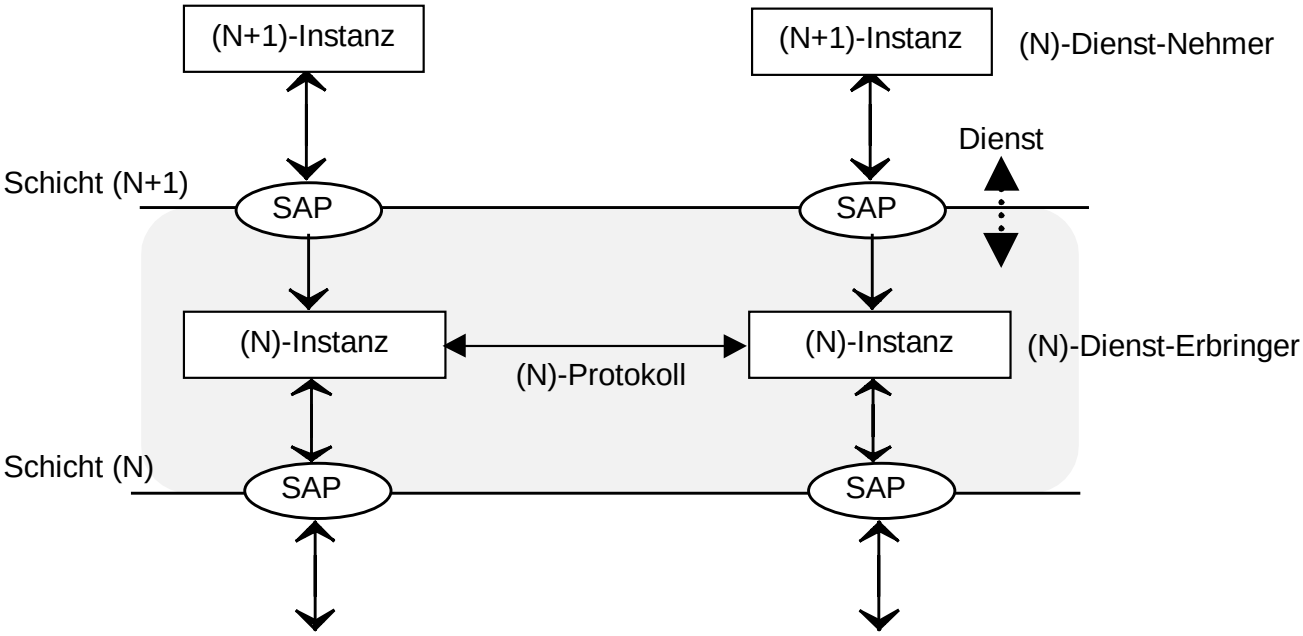



*Bild 2-7 Schichtung von Diensten*

Obwohl der Zwei-Benutzer-Dienst die am meisten vorkommende Art von Dienst ist, gibt es auch andere mögliche Dienste, z.B. erhalten bei einem **Multicast (Gruppendienst)** mehrere Empfänger Daten gleichzeitig, die von einem Benutzer gesendet werden.

Sowohl für die Kommunikationsprotokolle der einzelnen Schichten des Referenzmodells als auch für die von den Schichten erbrachten Dienste gibt es internationale Normen, die z.T. noch in der Entwicklung sind. Dienstnormen beschreiben die elementaren Interaktionen zwischen einem Benutzer und dem Erbringer des Dienstes an **Dienstzugangspunkten (SAP)**. Sie werden beschrieben in Form von **Dienstelementen (Primitives)**, die logisch als unteilbare Ereignisse angesehen werden. Dabei werden von OSI nur die Dienstelemente beschrieben, die sich auf die Kommunikation von Diensterbringer und Dienstbenutzer beziehen.

#### 2.3.2.1 Verbindungslose Dienste

Der verbindungslose Dienst ist gekennzeichnet durch die Übertragung beziehungsloser Dateneinheiten, sogenannten Datagrammen, von einem Quell-SAP zu einem oder mehreren Ziel-SAP's.

Da keine Verbindung aufgebaut wird, ist der Diensterbringer 'ohne Gedächtnis' über die Kommunikationsverbindung, es folgt daher keine Reihenfolgeprüfung und keine Fehlersicherung.

Verbindungslose Dienste gibt es in lokalen Netzen wie z.B. Ethernet, wo einfach Nachrichten gesendet werden, ohne dass eine Verbindung zwischen zwei Partnern aufgebaut wurde. Verbindungslose Dienste gibt es auch in Wide Area Networks. So gibt es beispielsweise bei DATEX-P einen Datagramm-Dienst. Hier werden auch Nachrichten abgeschickt, ohne dass erst eine Verbindung aufgebaut wurde. Diese Nachrichten enthalten neben den eigentlichen Daten den Absender und den Adressaten. Sie werden durch das Netz geschleust und beim Empfänger abgeliefert. Hierbei können auch Datagramme verloren gehen, das Netz bietet keine Sicherheit.

Bewertung von verbindungslosen Protokollen mittels Datagrammen für die Kommunikation über Fernmeldenetze:

+ einfaches Verfahren, kein Verbindungsaufbau erforderlich

- höhere Belastung der Vermittlung

- höherer Aufwand in DEE für Resequencing

Es ist möglich, dass auf einer datagramm-orientierten Netzverbindung (häufig bei LANs) ein verbindungsorientierter Dienst bereit gestellt wird.

#### 2.3.2.2 Verbindungsorientierte Dienste

Eine verbindungsorientierte Datenübertragung ist durch folgende Merkmale gekennzeichnet:

- Eindeutig bestimmte Lebensdauer

  (Verbindungsaufbau, Datentransfer, Verbindungsabbau)

- Verbindungskennung

  (Eine N-Verbindungsendpunktkennung wird auf beiden Seiten eines N-Dienstzugangspunkts (SAP) zur Identifikation der N-Verbindung benutzt.)

- Dreiseitige Übereinkunft. Sie umfasst Verhandlung zwischen:

  a) (N+1)-Instanz im System A und (N+1)-Instanz im System B

  b) (N+1)-Instanz im System A und (N)-Instanz im System A

  c) (N)-Instanz im System B und (N+1)-Instanz im System B

- Verhandlungen zwischen Instanzen. Verhandlungsgegenstände sind:

  a) notwendige Betriebsmittel (Puffer)

  b) Dienstgüteparameter (Durchsatz, Fehlerrate,...)

  etc.

Verbindungsorientierte Protokolle haben Mechanismen zur Erkennung fehlender Blöcke (Vollständigkeit nicht gewahrt), duplizierter Blöcke (Eindeutigkeit nicht gewahrt) und außerhalb der Sequenz eintreffenden Informationsblöcke (Sequenz nicht gewahrt) sowie zur Beseitigung der entsprechenden Fehlerzustände. Sie gewährleisten also Vollständigkeit, Eindeutigkeit und Sequenz.

### 2.3.3 Protokolle

[ Quelle : Helmut Kerner - Rechnernetze nach OSI ]

Die zweite Art von Beziehungen sind die horizontalen Beziehungen zwischen Instanzen, **Protokolle (protocols)** genannt. Das Wort "Protokoll" erinnert an Verabredungen zweier diplomatischer Vertretungen für die Durchführung von Staatsakten. Protokolle erkennt man auch in der gegenseitigen Begrüßung, in der man kaum ein frohes "Guten Morgen" mit "Auf Wiedersehen" beantworten sollte, wenn man ein Gespräch mit dem Partner beginnen möchte.

Auch beim Telefonieren folgt man einem Protokoll: wird man gerufen, so darf man sich mit "Hallo" melden; antwortet der Anrufende ebenfalls mit "Hallo", so befindet sich das Protokoll in einer Endlosschleife und man kommt nie zum eigentlichen Gespräch.

Protokolle sind Regeln zur Kommunikation zwischen Partnerinstanzen. In realen Systemen gibt es nur eine einzige physisch vorhandene horizontale Verbindung, nämlich jene durch das Übertragungsmedium. Da aber mittelbar - über einige vertikale Übergänge abwärts und dann über das Medium und jenseits des Mediums wieder aufwärts - auch eine Verbindung zwischen zwei **Partnerinstanzen (peer entities)** derselben Ebene existiert, kann man für die Kommunikation zwischen Partnerinstanzen diese als verbunden ansehen. Die Partnerinstanzen verkehren über diese Verbindung miteinander gemäß einem Protokoll ihrer Schicht.

Aufgaben von Protokollen:

| | |
|---|---|
| Adressierung | Adressangaben |
| Flusssteuerung | Quittungen, Empfangsfenster |
| Fehlererkennung | Prüfsumme, Sequenznummer, Quittungen, Zeitüberwachung |
| Fehlerbehebung | Paketwiederholung, Korrekturverfahren |

### 2.3.4 Paketstrukturen

[ Quelle : Helmut Kerner - Rechnernetze nach OSI ]

Es treten zwei Arten von Nachrichten auf, vertikal zwischen Folgeinstanzen laufende Nachrichten und horizontal zwischen Protokollpartnern auszutauschende Nachrichten.

Vertikal werden Dienstaufträge bzw. Zustimmungen zwischen der dienenden (z.B. (N)) und der fordernden Schicht (N+1) ausgetauscht. Sie bestehen aus einem **Kommandoteil, Interface Control Information, ICI** genannt, und aus **Nutzdaten** (**Userdaten**). Die Nutzdaten, welche der Schicht (N) übergeben werden, heißen **(N)-Service Data Unit, (N)-SDU**. Das übergebene Gesamtpaket (Kommandoteil plus Nutzdaten) heißt **Dienstelement** (**service primitive**).

Die Nutzdaten (user data), welche die (N+1)-Instanz ihrem Partner am anderen Ort senden will, muss sie der unteren Folgeinstanz im eigenen System zur Beförderung übergeben. Sie tut dies im (N)-SDU Feld des Übergabepakets. Diese Information soll von der (N)-Instanz unverändert an ihren (N)-Partner und weiter zur (N+1)-Instanz

befördert werden. Man bezeichnet die Übertragung daher als **transparent**, d.h. verborgen, da keine Information für die (N)-Instanzen darin enthalten ist.

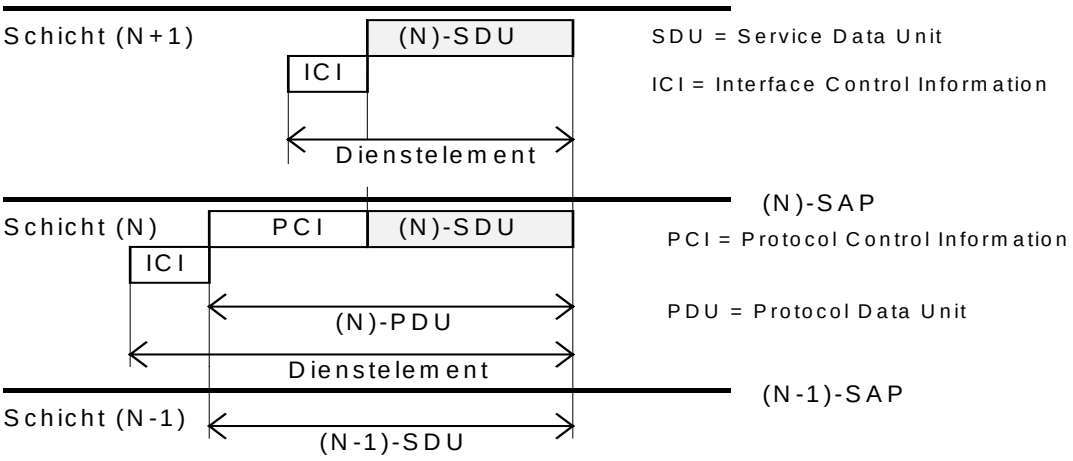



*Bild 2-8 Dateneinheiten aus Sicht der Schicht (N)*

Die für die (N)-Instanz bestimmte Steuerinformation ist davon getrennt in der ICI (Interface Control Information) gelagert. Dort befindet sich das Kommando an die Folgeschicht sowie weitere Steuerinformationen, wie z.B. der Zielort.

Die Inhalte und das Format der ICIs sind nicht genormt, da sie nur Übergänge innerhalb eines Systems betreffen und daher dem Hersteller des Systems überlassen bleiben. Die (N)-Instanz stellt nun jenes Informationspaket zusammen, das sein Protokollpartner (horizontal) erhalten soll, man nennt es **Protokolldateneinheit, ((N)-Protocol Data Unit, (N)-PDU)**. Es enthält in einem Feld unverändert die als (N)-SDU von oben übernommenen Daten (User data - Nutzdaten) und fügt noch Steueranweisungen an seinen Protokollpartner in einem eigenen Feld, dem **PCI-Feld (Protocol Control Information)** hinzu. Manche PDUs dienen nur reinen Steuerzwecken. Diese PDUs enthalten nur PCI-Informationen und in der Regel keine Userdaten (SDU).

Die (N)-PDU ist die Nutzinformation (Userdaten) der (N)-Schicht. Notgedrungen muss sie wieder an die nächst untere Folgeinstanz (N-1) zur Beförderung übergeben werden. Sie wird als (N-1)-SDU mit Steuerinformation (ICI) ergänzt und nach unten weiter gereicht.

Die eigentliche Nachricht wird in jeder Schicht mit einer Steuerinformation (PCI) für die Partnerinstanz versehen. In PDUs der untersten Schicht liegen somit 7 PCIs vor der ursprünglichen Nachricht. Nur die äußerste PCI ist für die unterste Schicht bestimmt. Den Vorgang könnte man damit vergleichen, dass ein Brief in ein Kuvert gesteckt wird, das wiederum in ein weiteres Kuvert verpackt wird, und das sieben mal. Jedes Kuvert enthält eine Anweisung für die Behandlung seines Inhalts.

Nachdem das Protokoll der Schicht 1 ausgeführt wurde, ist das äußerste Kuvert der Steuerinformation nicht mehr nötig. Bei jeder Weitergabe nach oben entfällt ein "PCI-Kuvert", bis zuletzt über der Schicht 7 nur noch die ursprüngliche Nachricht ankommt.

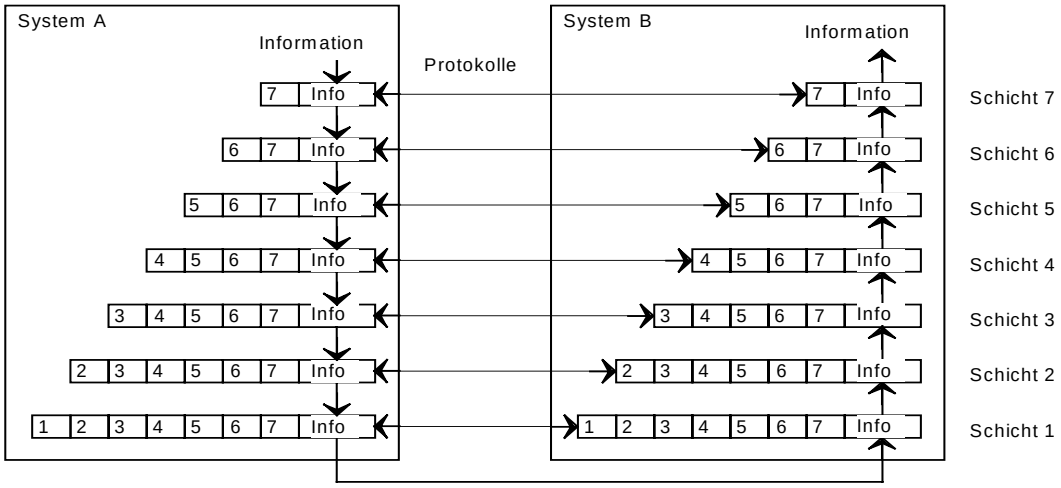



*Bild 2-9 Protokollweg durch den OSI-Turm*

## 2.4 Schichtaufgaben

[Quelle: Dieter Conrads, Datenkommunikation, Vieweg]

Gehen wir der Einfachheit halber von der historischen Situation aus (siehe Bild 2-10). Hierzu muss im Vorgriff gesagt werden, dass ein Rechner einer DEE alle 7 Schichten im Kommunikationssystem hat. Die Aufgabe einer DÜE ist zu übermitteln. Wie wir noch sehen werden, hat sie nur die unteren 3 Schichten des ISO/OSI-Modells, da Schicht 3 bereits die Vermittlung enthält.

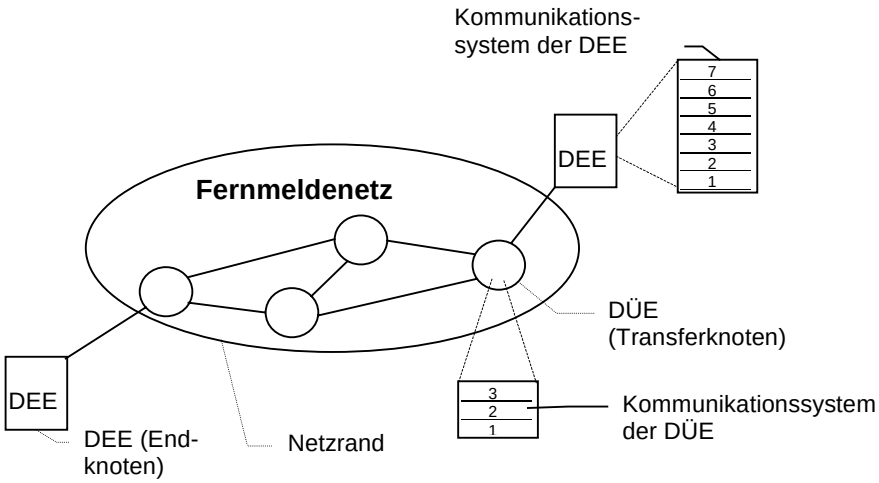



*Bild 2-10 Schichten für DEE[1] und DÜE[1]*

[1] Eine DEE ist ein Endsystem am Netzwerkrand, das mit anderen Endsystemen über das Netz kommuniziert. Die DÜE wird auch als Transitsystem oder Relais bezeichnet.

Das folgende Bild 2-11 zeigt einen Schnitt durch ein Netz mit nur einem Relais. Dargestellt sind

- das Kommunikationssystem einer DEE (linke DEE),
- das Kommunikationssystem der mit dieser DEE kommunizierenden DEE (rechte DEE),
- das Kommunikationssystem des Transitsystems (zwischen beiden DEEs)
- das Übertragungsmedium zwischen der linken DEE und dem Transitsystem
- und das Übertragungsmedium zwischen der rechten DEE und dem Transitsystem.

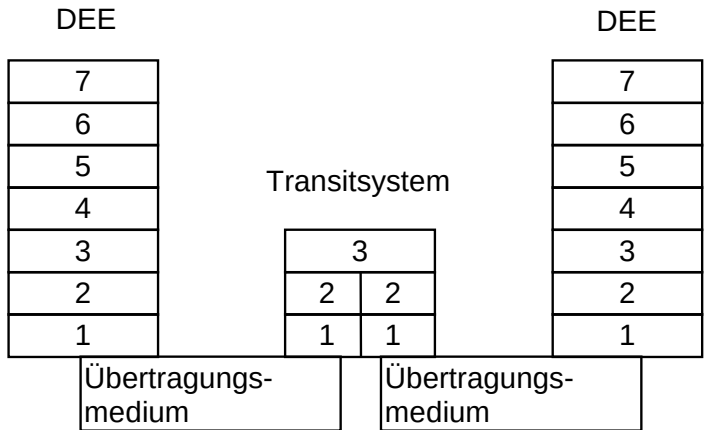



*Bild 2-11 Ein Transitsystem (Relais) hat 3 OSI-Schichten, ein Endsystem hat 7 Schichten*

Die Aufgaben der unteren 4 Schichten sind leicht verständlich. Mit den Aufgaben der oberen Schichten tut man sich etwas schwerer. Die Aufgaben der sieben Schichten sind folgende :

## 2.4.1 Bitübertragungsschicht (Schicht 1), physical layer

Die Schicht 1 beschreibt die physikalische Schnittstelle zum Übertragungsmedium. Zu der Spezifikation einer Schnittstelle zu einer Leitung gehört beispielsweise:

- die Spezifikation der Art der Leitung,
- die Spezifikation von Kabeln und Steckern,
- die Spezifikation der Übertragungstechnik (Modulation, elektrische Darstellung der Bits (Leitungscodes))
- etc.

Das Übertragungsmedium selbst gehört nicht zur Schicht 1.

Die Bitübertragungsschicht stellt ungesicherte Verbindungen zwischen Systemen für die Übertragung von Bits zur Verfügung. Als Fehlerfall kann nur der Ausfall der Verbindung gemeldet werden.

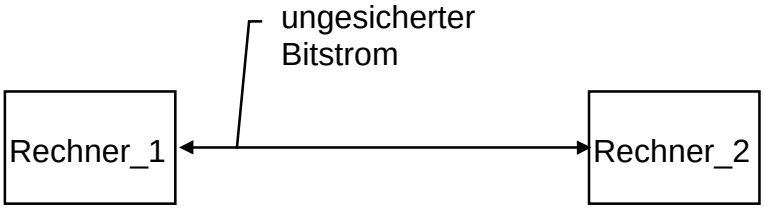



*Bild 2-12 Ungesicherter Bitstrom zwischen benachbarten Rechnern.*
*Rechner 1 sei etwa eine DEE, Rechner 2 eine DÜE*

### 2.4.2 Sicherungsschicht (Schicht 2), data link layer

Aufgabe der Schicht 2 ist es, den Verkehr zwischen zwei benachbarten Stationen, also einer Punkt-zu-Punkt-Verbindung, zu regeln. Die Schicht 2 hat 2 Aufgaben:

**Flusskontrolle** und **Fehlerbehandlung**.

**Flusskontrolle**

Wenn zwei unterschiedlich leistungsfähige Einheiten miteinander kommunizieren, muss die leistungsfähigere die Sendegeschwindigkeit so weit herabsetzen, dass die leistungsschwächere Einheit in der Lage ist, die Daten aufzunehmen. Über das Aussenden von Bestätigungen kann eine empfangende Station den Datenfluss steuern und dafür sorgen, dass sie sich nicht verschluckt.

**Fehlerbehandlung**

Die Informationen auf Schicht 2 sind kein kontinuierlicher Bitstrom, es sind Blöcke geeigneter Länge (frames oder Rahmen). Es ist nicht nötig, jeden Rahmen zu quittieren, sondern erst nach Empfang einer bestimmten Anzahl (Fenster-Technik: ein großes Fenster bedeutet, es wird erst nach einer großen Zahl von Messages quittiert, ein kleines Fenster bedeutet, es wird bereits nach einer kleinen Zahl von Messages quittiert.)

Ein Fehlercode in einem frame erlaubt die Fehlererkennung und -behebung (i.a. durch Wiederholung).

Bei lokalen Netzen erwies es sich als notwendig, die normalen Aufgaben der Schicht 2 in der Schicht 2b (Logical Link Control) unterzubringen. Schicht 2a wird benötigt für die Regelung des Zugriffs zum Medium (z.B. für das CSMA/CD-Verfahren im Falle von Ethernet). In lokalen Netzen hat man ja i.a. keine festen Punkt-zu-Punkt-Verbindungen zwischen direkt benachbarten Rechnern, da jede Station das Übertragungsmedium bekommen kann.


*Bild 2-13 Gesicherter Bitstrom zwischen benachbarten Rechnern. Rechner 1 sei etwa eine DEE, Rechner 2 eine DÜE*

Die Sicherungsschicht verbessert ungesicherte Verbindungen auf Teilstrecken zu gesicherten Verbindungen. Gesichert heißt dabei eine Verbindung, die Fehler in der Datenübertragung erkennt und korrigiert.

Mit anderen Worten, die Schicht 2 stellt eine gesicherte Punkt-zu-Punkt-Verbindung zwischen 2 Systemen zur Verfügung.

### 2.4.3 Vermittlungsschicht (Schicht 3), network layer

Die Vermittlungsschicht organisiert eine End-zu-End-Verbindung zwischen kommunizierenden Endsystemen.

Im Falle unseres Beispiels der Kommunikation zweier DEEs über ein Fernmeldenetz obliegt der Schicht 3 die Organisation der End-zu-End-Verbindung zwischen den beiden DEEs. Die Schicht 3 ist damit zuständig für die Wegewahl (das Routing), für das Multiplexen mehrerer Verbindungen über einzelne Teilstücke und für Aspekte der Fehlerbehandlung und Flusskontrolle zwischen den Endsystemen einer Verbindung. Mit anderen Worten, Schicht 3 kümmert sich um die End-zu-End-Verbindung zwischen kommunizierenden Rechnern und sorgt für den Transport von Paketen über die Teilstrecken des Netzes von Endsystem zu Endsystem.

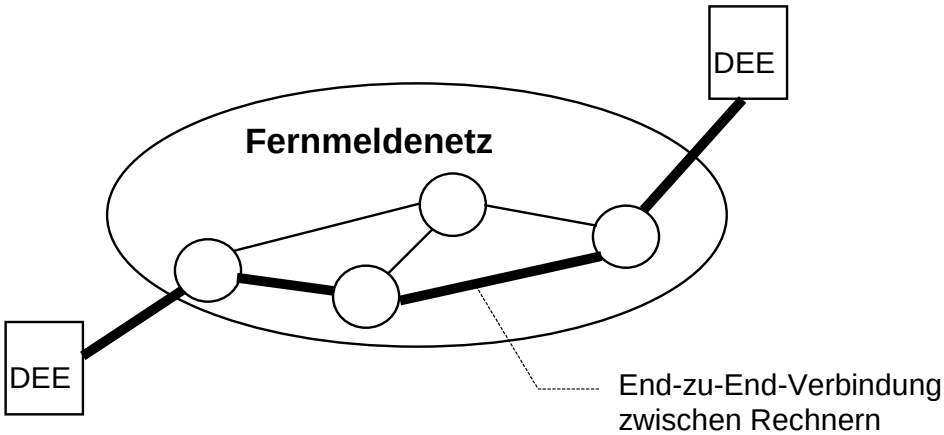



*Bild 2-14 End-zu-End-Verbindungen zwischen Rechnern*

Die Flusskontrolle auf der Schicht 3 macht die Flusskontrolle auf der Schicht 2 nicht überflüssig, da über eine Teilstrecke mehrere virtuelle Verbindungen, die zu verschiedenen End-zu-End-Verbindungen gehören, geführt werden können.

Die Fehlerbehandlung auf Schicht 3 bezieht sich nicht auf Übertragungsfehler - hierfür ist die Schicht 2 zuständig - sie bezieht sich auf Fehler, die beim Routing auftreten können. Hierzu gehört beispielsweise das Wiederherstellen der Sequenz, wenn Pakete in einer von der Sendefolge abweichenden Reihenfolge bei der Zielstation eintreffen.

Bei lokalen Netzen mit Datagramm-Dienst, bei denen also keine Verbindungen aufgebaut werden, ist die Schicht 3 praktisch funktionslos.

Es gibt weitere Ergänzungen zur Schicht 3, die das Internetworking zwischen verschiedenen Teilnetzen betreffen. Dies hat dazu geführt, dass die Schicht 3 in 3 horizontale Schichten neu eingeteilt wurde:

Schicht 3a (Subnetwork Access):

Wickelt die teilnetzspezifischen Protokolle (Routing etc.) ab.

Schicht 3c (Internet):

Wickelt die teilnetzunabhängigen Protokolle (Routing zu den Gateways zwischen Teilnetzen, falls Teilnetze verschiedene Protokolle fahren, globale Adressierung etc.) ab.

Schicht 3b (Subnet Enhancement):

ergänzt die Funktionen der Teilnetze so, dass die Anforderungen der Schicht 3c erfüllt werden.

Diese Einteilung kann an dieser Stelle nur erwähnt, nicht jedoch vertieft werden.

### 2.4.4 Transportschicht (Schicht 4), transport layer

| Die Transportschicht befasst sich mit End-zu-End-Verbindungen zwischen Prozessen in den End-Systemen. |
|---|

Sie wissen ja, dass jedes Endsystem parallel ablauffähige Einheiten (Prozesse) enthalten kann. Ein System kann beispielsweise 4 Prozesse enthalten, von denen jeder eine Verbindung zu einem Partner-Prozess in einem anderen Endsystem enthalten kann. Dabei ist es aber möglich, dass nicht alle 4 Verbindungen über dieselbe End-zu-End-Verbindung zwischen beiden Endsystemen gehen. Ein jeder Prozess fordert nämlich einen Transportdienst einer gewissen Güte an. Der Transportdienst ist transparent. Er verbirgt, welches Netz zum Transport überhaupt zum Zuge kommt. Es kann durchaus möglich sein, dass es aus Performance-Gründen (um die Güte des angeforderten Transportdienstes zu erfüllen), notwendig sein kann, mehrere End-zu-End-Verbindungen von End-System zu End-System aufzubauen. Die Daten werden dann gesplittet, über mehrere Schicht-3-Verbindungen geleitet und beim Empfänger in der Schicht 4 wieder zusammengefügt.

Natürlich muss sich die Transportschicht auch mit der **Flusskontrolle zwischen den kommunizierenden Prozessen** befassen.

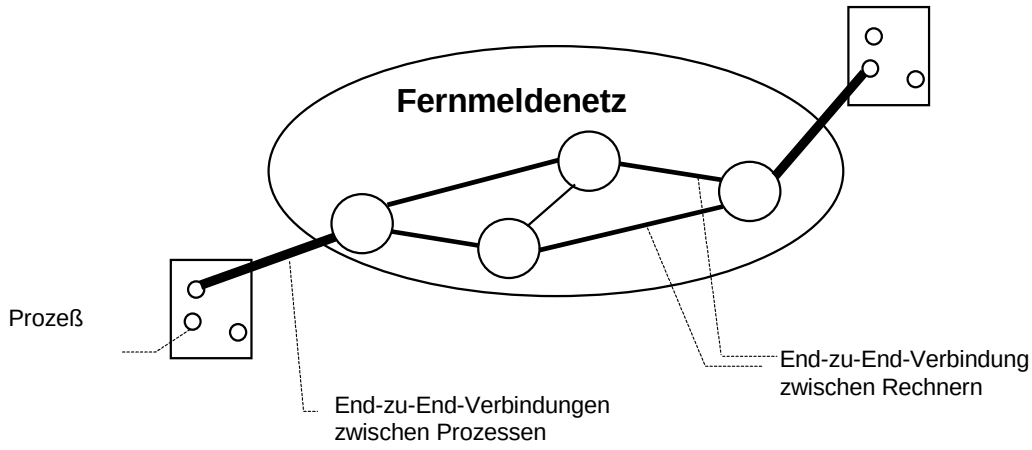



*Bild 2-15 Die Transportverbindung ━━━ zwischen zwei Prozessen kann auf mehrere Schicht 3-Verbindungen aufgesplittet werden.*

### 2.4.5 Kommunikationssteuerungsschicht (Schicht 5), session layer

Die Kommunikationssteuerungsschicht regelt die Synchronisation zwischen Prozessen in verschiedenen Systemen.

Sie ermöglicht z.B. den Wechsel der Gesprächsrichtung, wenn die Anwendung dies erfordert (der Transportdienst darunter kann etwa vollduplex sein). Sie ist auch verantwortlich für eine Resynchronisation in Fehlerfällen. Hierzu braucht sie vereinbarte Punkte im Datenstrom (Synchronisationspunkte), auf die erneut aufgesetzt werden kann. An dieser Stelle ist es nicht möglich, hierauf genauer einzugehen.

### 2.4.6 Darstellungsschicht (Schicht 6), presentation layer

Verschiedene Rechner stellen Daten verschieden dar. So kann es sein, dass einer der an der Kommunikation beteiligten Rechner keinen Datentyp String hat, dafür aber ein Array von Zeichen. Erhält er nun ein String-Objekt, so kann er damit nichts anfangen. Es stellt für ihn einen Fehler dar. Damit brauchen zwei Rechner, die miteinander kommunizieren wollen, wenigstens dieselben Datentypen. Werden Werte übertragen, so muss im Datenpaket auch verschlüsselt sein, zu welchem Typ sie gehören, damit sie richtig interpretiert werden. Dies bedeutet, dass man für den Datentyp eine **Transfersyntax** braucht.

Für die Kommunikation zwischen den Rechnern gibt es nun ein **„Esperanto“** für die Transfersyntax. **Teil 1 der Transfersyntax** betrifft die **Datentypen**. Die entsprechende „neutrale“ Syntax ist **ASN.1 (Abstract Syntax Notation 1)**. **Teil 2** des Esperantos sind die **Basic Encoding Rules for ASN.1**, die einen genormten Satz von **Regeln zur Codierung der Datenwerte in die Transfersyntax** beinhalten, damit für einen übertragenen Wert auch erkenntlich wird, zu welchem Typ er gehört.

Mit Hilfe dieser **neutralen Transfersyntax ASN.1/BER** (BER für Basic Encoding Rules) können nun Rechner miteinander Daten austauschen. Der Sender wandelt

vor der Datenübertragung seine lokale Syntax in die Transfersyntax. Der Empfänger wandelt wiederum die Transfersyntax in seine eigene lokale Syntax.

Kommunizierende Rechner können vereinbaren, dass sie als Transfersyntax nicht die neutrale Syntax nach ASN.1/BER nehmen, sondern eine Transfersyntax basierend auf der spezifischen lokalen Syntax, wenn beide diese spezifische lokale Syntax verstehen.

Die Aufgabe der Schicht 6 ist das **Aushandeln einer Transfersyntax**, die für beide Partner verständlich ist. Verstehen beide Rechner dieselbe lokale Syntax (z.B. Rechner desselben Herstellers) so können sie sich auf eine Transfersyntax auf Basis der lokalen Syntax einigen. Ist dies nicht der Fall, so müssen sie sich auf die neutrale Transfersyntax ASN.1/BER einigen. Die weiteren Aufgaben dieser Schicht sind die **Durchführung der Wandlung der lokalen Syntax in die Transfersyntax bzw. aus der Transfersyntax in die lokale Syntax**.

ASN.1 ist eine maschinenunabhängige Sprache für Datenstrukturen. ASN.1 ist eine formale Sprache mit einer eigenen Grammatik. ASN.1 ist die Netzwerk-Programmiersprache der neunziger Jahre. Die Basic Encoding Rules sind die Verpackungsregeln für die Verpackung der ASN.1-Werte in Nachrichten.

### 2.4.7 Anwendungsschicht (Schicht 7), application layer

Die Anwendungsschicht, die höchste Schicht des Referenzmodells, stellt die Schnittstelle der Anwendungsprozesse zum Kommunikationssystem dar. Es handelt sich hierbei um allgemeine Hilfsdienste für die Kommunikation oder aber spezielle Kommunikationsdienste wie File Transfer, Message Handling System (MHS), etc.

Der Anwendungsprozess hat ausschließlich über die Schicht 7 Zugang zum OSI-Kommunikationssystem.

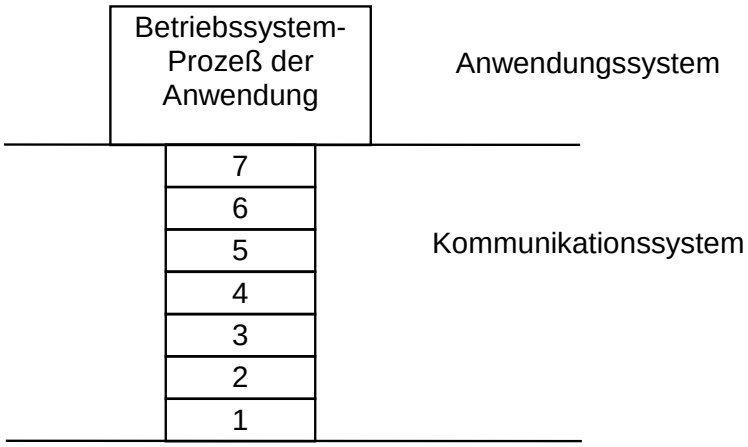



*Bild 2-16 Ein Anwendungsprozess (hier: ein Betriebssystem-Prozess der Anwendung) kann auf das Kommunikationssystem nur über die Dienste der Schicht 7 des Kommunikationssystems zugreifen*

## 2.5 Aufgaben

1 a) Wer stellt wem einen **Dienst** im Rahmen von ISO/OSI zur Verfügung ?

b) Durch wen werden die Leistungen des Dienstes erbracht ?

c) Benennen Sie in der Terminologie von ISO/OSI die Arbeitseinheit, die den Dienst anbietet.

d) An welcher Stelle bietet die Arbeitseinheit den Dienst an ?

e) Regelt ISO/OSI die Implementierung des Aufrufs eines Dienstes ?

2 a) Was ist ein Dienstelement ?

b) Wozu dient eine Interface Control Information ?

c) Wozu dient eine Protocol Control Information ?

3 a) Welche der Schichten Hardware, Betriebssystem, Kommunikationssystem, Anwendungssystem wird durch das ISO/OSI-Modell modelliert ?

b) Wo gehört im ISO/OSI-Schichtenmodell ein Anwendungsprozess (z.B. ein Flugbuchungssystem) hin ?

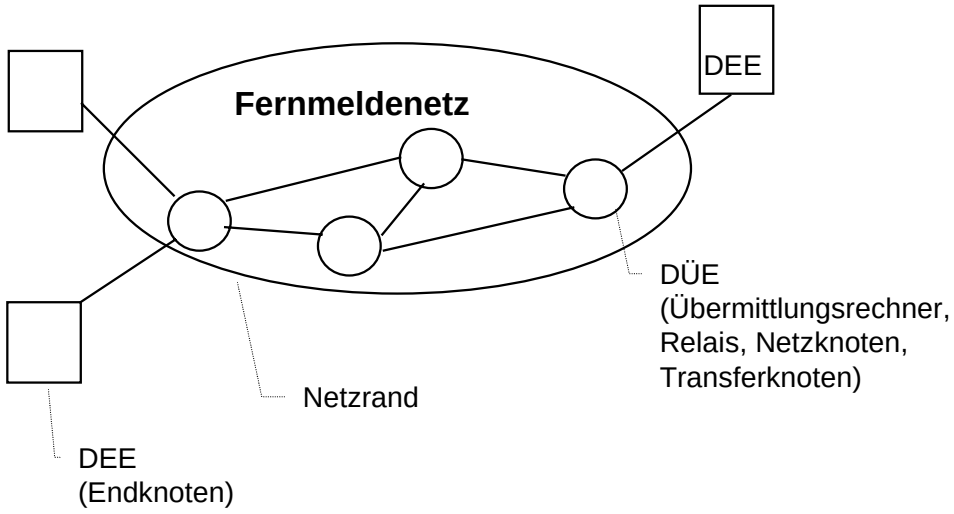
4 a) Was ist ein Protokoll ?

b) Welche Aufgaben hat ein Protokoll und welche Realisierungs-Möglichkeiten gibt es zur Durchführung dieser Aufgaben ?

5 a) Was versteht man unter Flusskontrolle ?

b) Welche Aufgabe hat die Flusskontrolle auf ISO/OSI-Schicht 2, auf Schicht 3 und auf Schicht 4 ?

c) Zeigen Sie an einem Beispiel, dass man eine Schicht 2-Flußkontrolle braucht, obwohl eine Schicht 3-Flußkontrolle vorhanden ist. Skizze!



6 Geben Sie für die Fragen a) bis h) jeweils an, welche ISO/OSI-Schicht dafür zuständig ist:

a) Resynchronisation von Prozessen

b) Flusskontrolle von End-zu-End-Verbindungen zwischen Prozessen

c) Aufbau von End-zu-End-Verbindungen zwischen Systemen

d) Spezifikation der Übertragungstechnik

e) Erkennen des Bruchs der Leitung

f) Wechsel der Gesprächsrichtung

g) Erkennen falsch gerouteter Pakete

h) Erkennen einer fehlerhaften Bit-Übertragung

7 a) Beschreiben Sie die Aufgaben von Schicht 5 des ISO/OSI-Referenzmodells.

b) Beschreiben Sie die Aufgaben von Schicht 4 des ISO/OSI-Referenzmodells.

c) Beschreiben Sie die Aufgaben von Schicht 3 des ISO/OSI-Referenzmodells.

8 Was versteht man unter Peer-to-Peer Kommunikation ?

# 3 Netzkoppelelemente

## 3.1 Einleitung

Um die Verbindung zwischen zwei Netzen herzustellen, die aufgrund technischer oder geographischer Gegebenheiten nicht direkt (also durch ein Kabel) gekoppelt werden können, werden Netzkoppelelemente (auch Netzkoppeleinheiten genannt) verwendet. Diese sind im allgemeinsten Fall Stationen an zwei Netzen und stellen durch ihre Funktion die Verbindung zwischen diesen Netzen her.

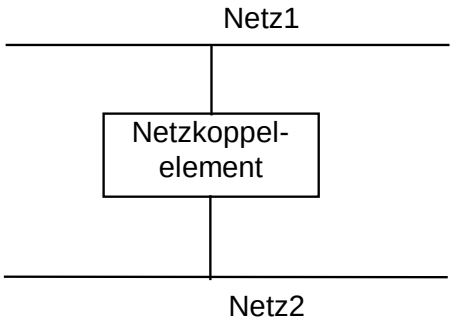



*Bild 3-1 Netzkoppelelement*

Durch Netzkoppeleinheiten ist es möglich

- die physikalische Begrenzung der Ausdehnung eines lokalen Netzes (durch **Kopplung mehrerer** solcher **Teilnetze**) zu umgehen
- Netze mit **unterschiedlichen Protokollarchitekturen** miteinander zu koppeln (z.B. Netze mit herstellerspezifischen Protokollen wie IBM-SNA oder DECnet)
- lokales Datenaufkommen durch Bildung von **logische Teilnetzen** vom restlichen Netzwerk zu entkoppeln
- Teilnetze mittels Netzkoppeleinheiten zu einem Netz für den bereichs-übergreifenden Verkehr zu koppeln, damit nur noch bereichsübergreifender Datenverkehr das gesamte Netz belastet (**Lastentkopplung**).

Bevor auf die Aufgaben von Netzkoppeleinheiten näher eingegangen wird, sollen hier noch einige häufig benötigte Begriffe eingeführt werden:

**Teilnetz**

Rechner, die über eine gemeinsame Verbindungsschicht (Schicht 2 des OSI-Referenzmodelles) erreicht werden, formen aus Sicht der Netzwerkschicht ein Teilnetz. Der Einsatz der Netzwerkschicht ist in der Regel nur dann sinnvoll, wenn Rechner in verschiedenen Teilnetzen miteinander kommunizieren. ([2], S.75-76)

**Heterogenes Netz**

Unter einem heterogenen Netz versteht man ein Rechnernetz, das aus heterogenen Komponenten aufgebaut ist.

Heterogene Komponenten sind gekennzeichnet durch:

- verschiedene Zugriffsverfahren:
  Es kommen verschiedene Zugriffsverfahren zum Einsatz, beispielsweise Zufallsverfahren (z.B. Ethernet) und geregelte Zugriffsverfahren (z.B. Token Ring).
- verschiedene Protokolle und Netzwerkbetriebssysteme:
  Im Netzwerk werden verschiedene Protokolle und Netzwerkbetriebssysteme eingesetzt. Zum Beispiel könnten in einem Netzwerk die Rechner unter Verwendung von DECnet und TCP/IP vernetzt sein.
- verschiedene angeschlossene Rechner:
  Die angeschlossenen Rechner unterscheiden sich nach Rechnerarchitektur (Prozessor), Rechnerklasse (PC, Mainframe,...), Betriebssysteme, Dateisysteme und Hersteller.

([11], S. 41-42)

In nachfolgender Abbildung ist ein Beispiel für ein heterogenes Netz zu sehen:

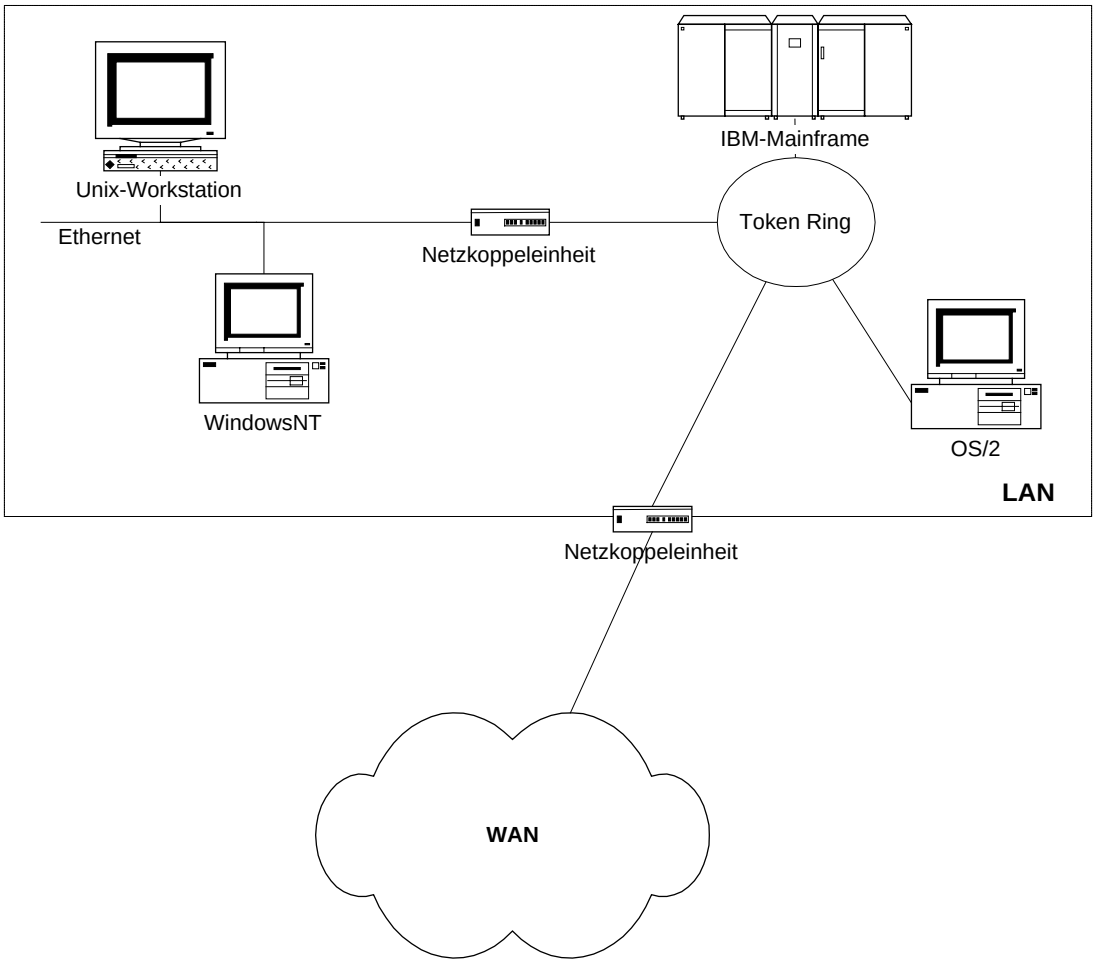



*Bild 3-2 Heterogenes Netz*

**Backbone Netz**

Sollen mehrere Netze (unterschiedlicher Protokollarchitekturen) gekoppelt werden, so kann es aus Gründen der Erweiterbarkeit und der Leistungsfähigkeit sinnvoll sein, diese nicht direkt über ein Netzkoppelelement, sondern indirekt über ein Backbone-Netz miteinander zu koppeln.

Bei einem Backbone-Netz handelt es sich um ein Netz mit einem von den angeschlossenen Stationen unabhängigen, standardisierten Protokoll. Backbone-Netze sind in der Regel Netze mit einer sehr hohen Datentransferrate (z.B. FDDI, ATM), da die Gesamtleistung des Netzes in einem hohen Maß von der Leistungsfähigkeit des Backbones abhängt.

Beim Einsatz eines Backbone-Netzes müssen zwar pro Informationsaustausch zwei Protokollumsetzungen vorgenommen werden (es sind somit mindestens zwei Netzkoppeleinheiten erforderlich), dafür reduziert sich jedoch die Anzahl der denkbaren Protokollumsetzungsvarianten.

Nachfolgende Abbildung zeigt den prinzipiellen Einsatz eines Backbone-Netzes:

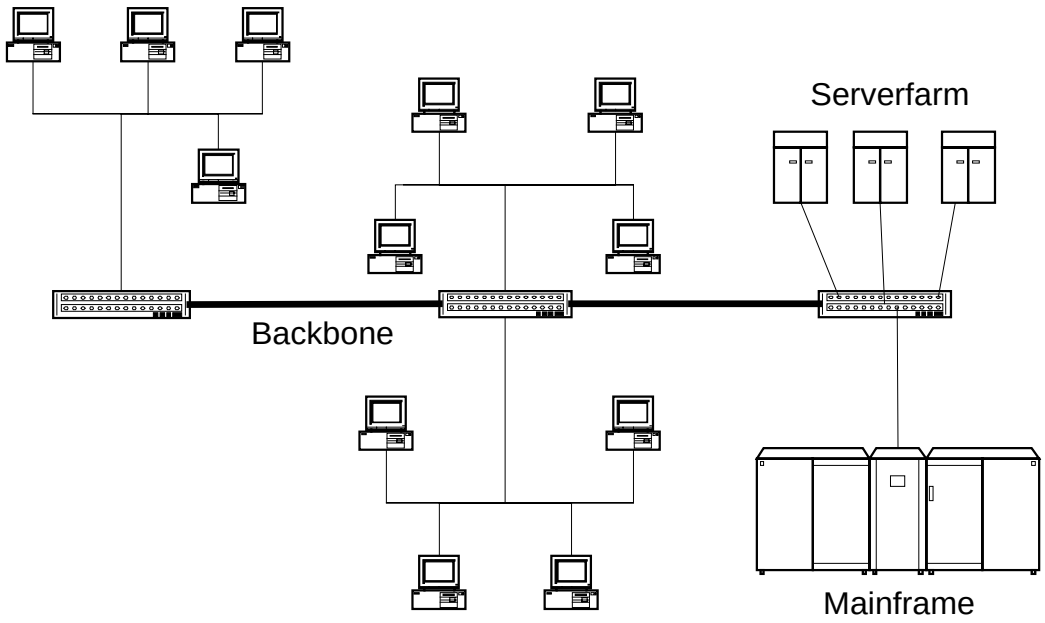



*Bild 3-3 Backbone-Netz*

Der in *Bild 3-3* gezeigt „klassische“ Backbone kommt in modernen Netzen nur noch dann zum Einsatz, wenn Teilnetze in mehreren Gebäuden miteinander verbunden werden müssen. Befindet sich das Netz innerhalb eines Gebäudes, wird zumeist an zentraler Stelle ein leistungsfähiges Koppelelement installiert, an das sternförmig weitere Koppelelemente oder auch einzelne Rechner angeschlossen werden.

Da sich der Backbone dann quasi innerhalb des Koppelelementes befindet, spricht man von einem „collapsed Backbone“. Innerhalb des Koppelelementes sind Übertragungsraten von mehreren GBit/s möglich, was einen entsprechend hohen Durchsatz zur Folge hat. Die Verwendung eines zentralen Koppelelementes hat jedoch den Nachteil, dass eine Fehlfunktion dieses Gerätes den gesamten Datenverkehr im Netz zum Erliegen bringt; hier sind Maßnahmen zur Absicherung

des Gerätes (redundante Ausführung der wichtigsten Komponenten oder Verwendung eines zweiten Koppelelementes parallel zum ersten) notwendig.

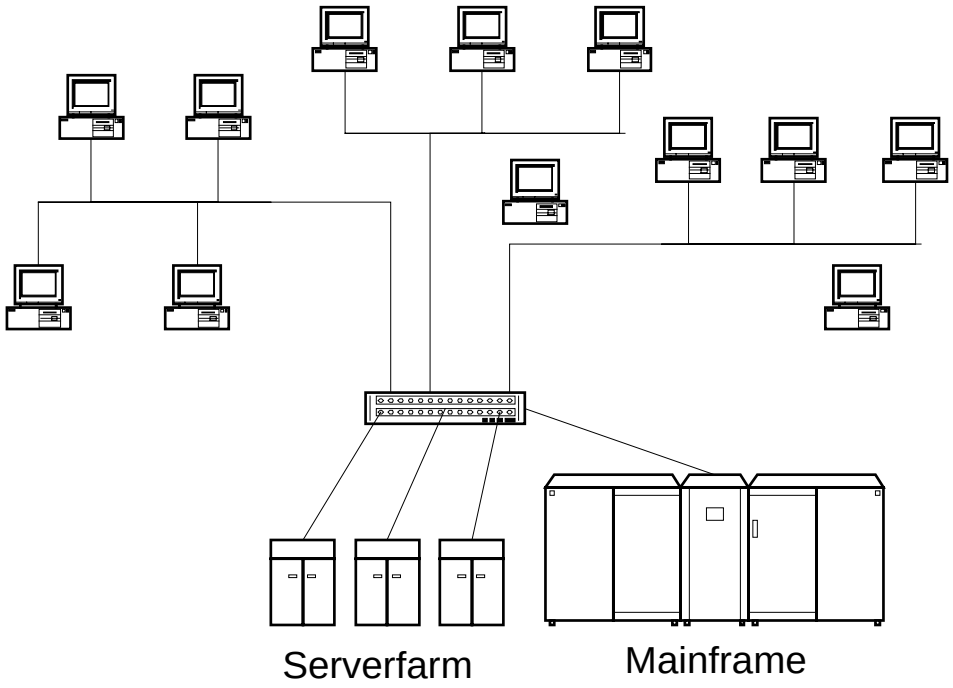



*Bild 3-4 Collapsed Backbone*

## 3.2 Aufgabe einer Netzkoppeleinheit

Zu koppelnde Netze können sich in vielen Merkmalen unterscheiden. Es ist Aufgabe der Netzkoppeleinheit, diese Unterschiede zu überwinden und eine Kommunikation zu ermöglichen.

In einer Netzkoppeleinheit werden die Protokolle der beteiligten Netze aufeinander abgebildet. Handelt es sich bei den beteiligten Netzen um Netze mit gleicher Protokollarchitektur, so kann die Information direkt weitergegeben werden. Handelt es sich allerdings um Netze **unterschiedlicher Protokollarchitektur**, so müssen bei einem Informationsaustausch von einem zum anderen Netz die **Steuerinformationen des einen Netzes entfernt** und die **Steuerinformationen des anderen Netzes eingefügt bzw. eingepackt** werden.

In den folgenden Abschnitten sind die wesentlichen Aufgaben der Netzkoppel-einheiten aufgeführt, wobei die unterschiedlichen Netzkoppelsysteme ein unterschiedliches Maß an Aufgaben zu erfüllen haben. Dies hängt davon ab, wie stark sich die zu koppelnden Netze unterscheiden. Beispielhaft sei hier der **Repeater** erwähnt, der lediglich dazu benutzt wird, **Netzsegmente gleichen Typs** miteinander zu **koppeln**. Es ist leicht einzusehen, dass seine Aufgaben lange nicht so weitreichend sind wie z.B. die eines Gateways, das zur Kopplung von Netzen mit unterschiedlicher Protokollarchitektur dient.

### 3.2.1 Abbildung der PDU-Parameter

Die Sequenzen von Protokolldaten (PDUs, Protocol Data Units) des einen Netzes müssen auf Sequenzen von PDUs des anderen Netzes abgebildet werden. Dabei werden die Parameter von PDUs wie folgt unterschieden:

- **Parameter, die in beiden Netzen dieselbe Bedeutung haben** und somit nicht umgesetzt werden brauchen.
- **Parameter, die umgesetzt werden müssen, da sie in beiden Netzen unterschiedliche Bedeutung haben**.
- **Parameter, die nur im sendenden Netz definiert sind** und somit von der Netzkoppeleinheit entfernt werden müssen.
- **Parameter, die nur im empfangenden Netz definiert sind** und somit von der Netzkoppeleinheit erzeugt werden müssen. ([1], S.3-4)

Zur Erinnerung soll an dieser Stelle noch einmal der Begriff **'Protocol Data Unit'** wiederholt werden. Eine PDU ist ein Datenpaket, das sich aus den eigentlich zu versendenden Benutzerdaten, die als Service Data Unit (SDU) bezeichnet werden, und einem kleinen Kopf, der die zu übertragenden Benutzerdaten identifiziert, zusammensetzt. Dieser Kopf wird als Protocol Control Information (PCI) bezeichnet und beinhaltet Information, die zwischen zwei Einheiten auf der gleichen Schicht N ausgetauscht wird, um ihre gemeinsame Operation zur Erbringung von Dienstleistungen zu koordinieren:

### 3.2.2 Anpassung der PDU-Größen

Zusätzlich zur Umsetzung der unterschiedlichen Parameter müssen auch die **Größen der PDUs** einander **angepasst** werden. Dabei treten Probleme auf, wenn in Netz 1 die minimale Größe von Netz 2 unterschritten wird, bzw. wenn in Netz 1 die maximale Größe von Netz 2 überschritten wird.

Das erste Problem lässt sich durch **Verkettung (Concatenation)** von mehreren PDUs zu einer PDU lösen. Dabei muss beachtet werden, dass die **einzelnen PDUs im Empfänger** durch Aufspalten (Separation) der zusammengesetzten PDU **wieder zurückgewonnen** werden müssen. Eine andere Möglichkeit ist das Hinzufügen von Füllbits, die allerdings beim Empfänger wieder als solche erkannt und entfernt werden müssen.

Das zweite Problem lässt sich durch **Aufspalten der PDU in kleinere PDUs (Segmenting)** lösen, sofern das Protokoll auf der Kopplungsschicht diese Funktion vorsieht. **Spätestens beim Empfänger** müssen die zusammengehörenden Segmente wieder zusammengestellt und zu der ursprünglichen PDU **zusammengefügt** werden **(Reassemble)**. ([1], S.9-10)

### 3.2.3 Anpassung der Übertragungsgeschwindigkeiten

Beim Umsetzen der PDU-Sequenzen spielen auch die unterschiedlichen Übertragungsgeschwindigkeiten eine Rolle. Diese müssen durch die Netzkoppeleinheit einander angepasst werden.

Bei **durchschaltevermittelnden Netzen** wird die Anpassung der Übertragungsgeschwindigkeit realisiert durch Mechanismen wie

- **Multiplexen von Verbindungen (Multiplexing)** bei der Kopplung von einem langsamen auf ein schnelles Netz,
- und **Aufspalten von Verbindungen (Splitting)** bei der Kopplung von einem schnellen auf ein langsames Netz.

Bei **paketvermittelnden Netzen** gibt es mehrere Möglichkeiten. Beim **Übergang von einem langsamen auf ein schnelles Netz** entstehen in der Regel keine Probleme. Die Kapazität des schnellen Netzes wird dann oft nicht voll ausgenutzt, insbesondere bei Eins-zu-Eins-Abbildungen. In der umgekehrten Richtung beim **Übergang von einem schnellen auf ein langsames Netz** ist, **falls keine Flusskontrolle vorhanden** ist, in der Netzkoppeleinheit ein **sehr großer Pufferspeicher** notwendig, um die empfangenen PDUs zwischenzuspeichern, bevor sie auf dem langsamen Netz wieder ausgesendet werden können. Läuft der Pufferspeicher über, gehen die PDUs verloren.

Eine weitere Möglichkeit ist die eben erwähnte **Flusskontrolle**, welche bei verbindungsorientierten Protokollen möglich ist. Eine End-zu-End-Flusskontrolle oberhalb der Koppelschicht passt die Geschwindigkeit von Sender und Empfänger einander an, kann aber natürlich nicht verhindern, dass die **Netzkoppeleinheit eventuell durch die Summe der Verbindungen überlastet** wird. Separate Flusskontrollen auf den beiden Übertragungsabschnitten erlauben die Anpassung von Sende- und Empfangsgeschwindigkeit auf dem jeweiligen Übertragungsabschnitt. Jedoch erst die Kopplung dieser Flusskontrollen hilft, einen großen

Pufferspeicher in der Netzkoppeleinheit zu vermeiden, indem sich der Rückstau bereits beim Sender ergibt, falls Empfänger oder Netzkoppeleinheit die PDUs nicht schnell genug bearbeiten können. ([1], S.8-9)

### 3.2.4 Wegsuche (Routing) und Adressierung

Mittels Routing sollen Pakete effizient vom Absender zum Ziel zugestellt werden. Üblicherweise werden mit diesem Begriff zwei separate Funktionen zusammengefasst:

- Beschaffung und Unterhaltung von Zustellinformationen
- Ausnutzung der Zustellinformationen für die Zustellung

Unter **Zustellinformation** versteht man z.B. Weglängen, Adressen, usw. Die **Routing-Entscheidung** ('welchen Weg wähle ich?') kann auf **zwei Kriterienklassen** beruhen.

Die **erste Klasse** beinhaltet **fixe Kriterien**, wie zum Beispiel Wege oder Teilwege zu Zielen. Diese werden manuell in statische Routing-Tabellen eingetragen. Die Nachführung dieser Tabellen ist lästig, allerdings hat diese Variante Sicherheitsvorteile, da nur die bekannten eingetragenen Adressen transportiert werden und andere Stationen keine Möglichkeit haben, in die Kommunikation über die Netzkoppeleinheit hinweg eingebunden zu werden. Man spricht bei diesem Verfahren deshalb auch vom sogenannten **Protected Mode**. Dynamische Änderungen im Netz, wie z.B. das Ein- und Ausschalten eines Systems werden nicht berücksichtigt.

Die **zweite Klasse** beinhaltet **dynamische Kriterien**, die teilweise berechnet werden und in einer dynamischen Tabelle stehen. Hier werden Änderungen im Netz erkannt und entsprechend vermerkt. Zu den Kriterien zählen z.B.:

- momentan arbeitende Knoten
- minimale Anzahl von passierten Knoten
- minimale Weglänge zu einem Knoten
- Gesamtlast
- gleichmäßige Auslastung der Ressourcen
- Verfügbarkeit der Verbindung

### 3.2.5 Übergang verbindungsorientiertes - verbindungsloses Netz

Beim Übergang von einem verbindungsorientierten auf ein verbindungsloses Netz können auf Grund der fehlenden Kanalreservierungen im verbindungslosen Netz Verluste von PDUs auftreten, die der Sender oder die Netzkoppeleinheit erkennen und ausgleichen muss.

Beim Übergang von einem verbindungslosen auf ein verbindungsorientiertes Netz muss die Netzkoppeleinheit bei jeder PDU überprüfen, ob das gewünschte Ziel über eine bereits bestehende Verbindung erreicht werden kann. Wenn keine solche

Verbindung existiert, muss die Netzkoppeleinheit die PDU zwischenspeichern, die benötigte Verbindung aufbauen und erst dann die PDU über diese Verbindung übertragen.

Durch die fehlende Flusskontrolle auf dem verbindungslosen Netz ist in der Netzkoppeleinheit ein relativ großer Pufferspeicher notwendig, der die PDUs aufnehmen kann, die auf einen Verbindungsaufbau oder auf die Sendeerlaubnis durch die Flusskontrolle des verbindungsorientierten Netz warten. ([1], S.10-11)

## 3.2.6 Prinzipielle Realisierungsmöglichkeiten

Geht man bei der Kopplung zweier Netze davon aus, dass auf der Schicht N-1 die Protokolle der zu koppelnden Netze verschieden und von Schicht N an aufwärts identisch sind (die Protokolle der Schichten 1 bis N-2 können, soweit vorhanden, identisch oder verschieden sein), so gibt es folgende Möglichkeiten der Kopplung:

- Netzkopplung durch Protokolltransformation
- Netzkopplung durch Einführen eines globalen Protokolls
- Netzkopplung auf der ersten gemeinsamen Schicht

### 3.2.6.1 Netzkopplung durch Protokolltransformation

Die Kopplung durch Protokolltransformation, die in nachfolgender Graphik dargestellt ist, wird auf der letzten unterschiedlichen Schicht (N-1) realisiert. Sie hat oft einen Verlust der Funktionalität zur Folge, da zu Funktionen des einen Netzes nicht immer eine entsprechende Funktion auf dem anderen Netz existiert. ([1], S.4-5)

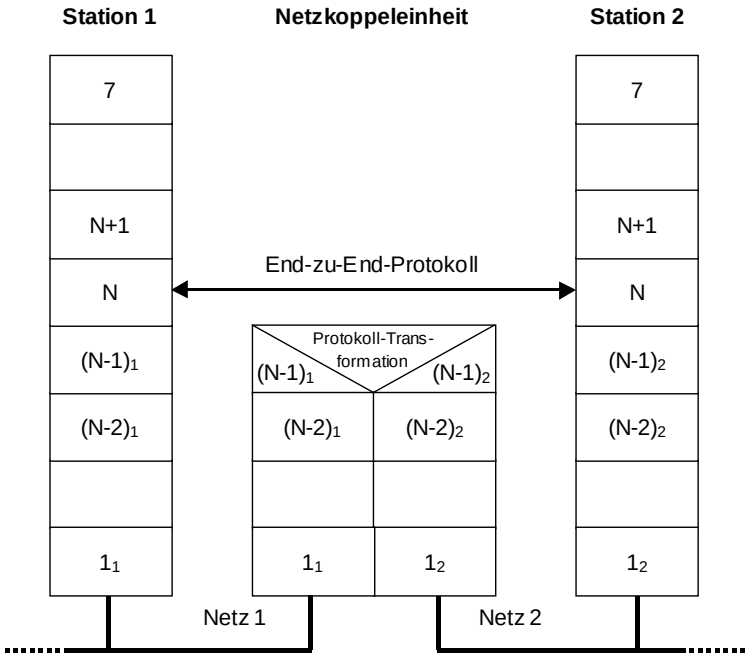



*Bild 3-5 Netzkopplung durch Protokolltransformation*

### 3.2.6.2 Netzkopplung durch Einführung eines globalen Protokolls

Auch die Kopplung durch Einführung eines globalen Protokolls wird in der letzten verschiedenen Schicht (N-1) realisiert. In dieser Schicht wird ein globales Protokoll sowohl in Netz 1 als auch in Netz 2 eingeführt, in welches die netzspezifischen Protokolle dieser Schicht mittels einer Anpassungsschicht angepasst werden. Diese Kopplungsart, welche nachfolgend dargestellt ist, hat den Nachteil, dass die Protokolle der Schicht N-1 in allen Stationen der beteiligten Netze erweitert werden müssen. ([1], S.5-6)

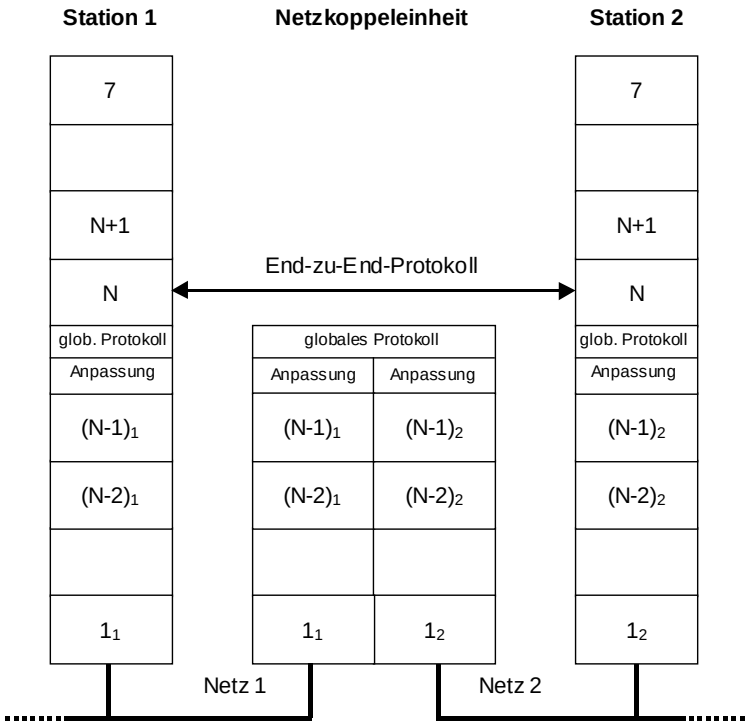



*Bild 3-6 Netzkopplung durch Einführung eines globalen Protokolls*

Der Vorteil der beiden oben aufgeführten Kopplungsarten sind die End-zu-End-Protokolle von der Schicht N an aufwärts, wodurch z.B. eine End-zu-End-Flusskontrolle realisiert werden kann.

### 3.2.6.3 Netzkopplung auf der ersten gemeinsamen Schicht

Nachfolgende Graphik stellt eine Kopplung auf der ersten gemeinsamen Schicht dar. Hier ist keine Protokolltransformation notwendig. PDUs werden von der einen Seite der Netzkoppeleinheit in der Schicht N auf die andere Seite weitergegeben. Ein Erweitern der Protokolle ist nicht notwendig, dafür geht das End-zu-End-Protokoll auf der Schicht N verloren. ([1], S.6)

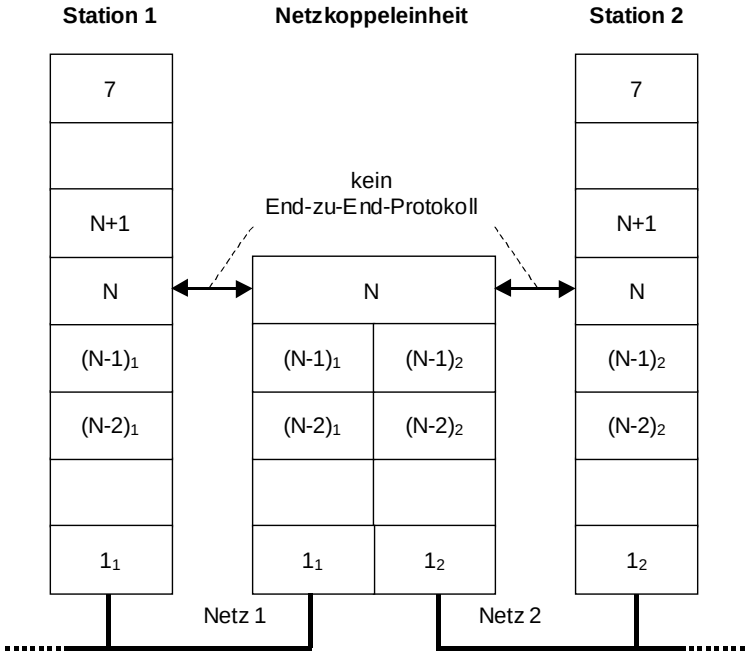



*Bild 3-7 Netzkopplung auf der ersten gemeinsamen Schicht*

## 3.3 Repeater

Repeater stellen die einfachste Art der Netzkopplung dar. Sie verbinden zwei Netze gleichen Typs (im allgemeinen zwei Ethernet-Segmente) auf der Bitübertragungsschicht (Schicht 1). Der Repeater ist ein elektrischer Signalverstärker und dient dazu, Netze, deren räumliche Ausdehnung aus physikalischen Gründen (Signaldämpfung und Signalverformung, z.B. durch Dispersion) begrenzt sind, zu erweitern.

Die Anzahl der zu koppelnden Netze ist dabei begrenzt, da es bei den einzelnen lokalen Netzen Einschränkungen der Ausdehnung durch das Übertragungsprotokoll gibt. Das bei Ethernet verwendete Zuteilungsverfahren CSMA/CD (Carrier Sense Multiple Access / Collision Detection) beispielsweise schreibt eine maximal erlaubte Laufzeit der Daten vor, um die sichere Erkennung von Kollisionen zu gewährleisten. Zu einer Kollision kommt es, wenn zwei Stationen gleichzeitig auf das Übertragungsmedium zugreifen um eine Nachricht zu versenden. Durch den Kollisionserkennungsmechanismus wird die Kollision von beiden sendenden Stationen erkannt, worauf diese ihren Sendebetrieb einstellen und ihn nach kurzer Pause zeitlich versetzt wieder aufnehmen. Die sendenden Stationen hören das Medium nur solange ab, solange sie sich im Sendebetrieb befinden. Wird innerhalb dieser Zeit keine Kollision erkannt, so wird dies von der sendenden Station als Ankunftsbestätigung aufgefasst. Das Signal muss also innerhalb der kürzesten möglichen Paketdauer bis an das Ende des Netzes und wieder zurück gelangen, damit der Mechanismus funktioniert. Aus diesem Grund dürfen sich bei Ethernet zwischen zwei beliebigen Endstationen maximal 4 Repeater befinden.

Bei der Kopplung mittels Repeatern müssen beide miteinander zu verbindende Netzwerke auf allen sieben Schichten des OSI-Referenzmodells identisch sein. Da Repeater die Schichten 2 bis 7 nicht auswertet, ist es gleichgültig, welche Netzwerksoftware und welches Netzwerkbetriebssystem eingesetzt wird.

Ein Repeater kopiert blindlings jedes Bit von einem Kabelsegment zum anderen, durch ihn erhält man praktisch ein Netz, das aus mehreren Netzsegmenten aufgebaut ist. D.h. durch den Einsatz eines Repeaters erfolgt keine Aufsplittung des Netzes in logische Teilsegmente, wie dies zum Beispiel bei Bridges der Fall ist. ([1], S.13-14; [3], Kap. 20.1; [4], S.50-51; [7], S.15)

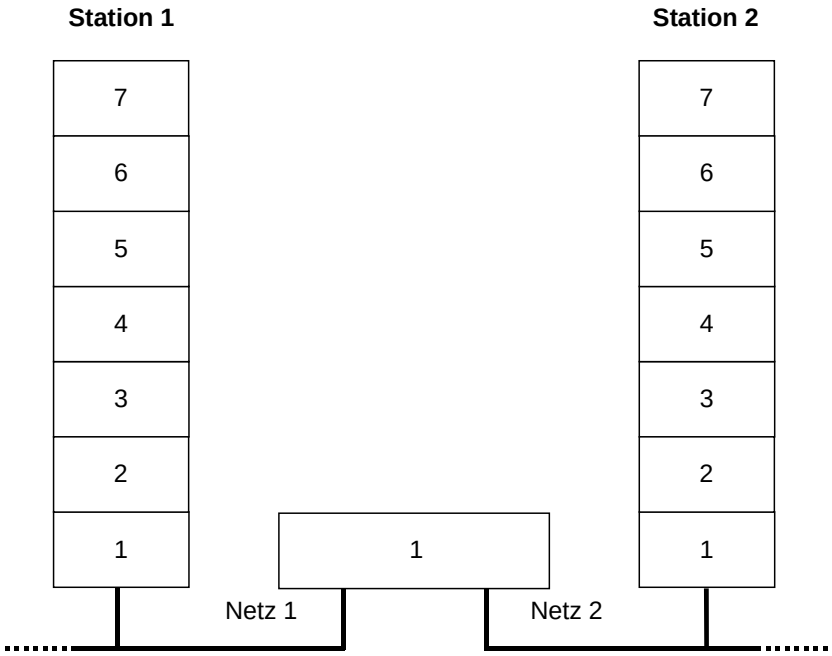



*Bild 3-8 Netzkopplung mit einem Repeater*

Neben dem herkömmlichen Repeater, der genau zwei Netze miteinander koppelt entstanden zur Kopplung mehrerer Netze, bzw. zur Kopplung von räumlich weit voneinander entfernten Netzen, folgende Weiterentwicklungen:

### 3.3.1 Multiport Repeater

Multiport Repeater stellen eine besondere Repeater-Bauart dar. Multiport-Repeater haben mehrere LAN-Ports und **können** so **mehrere Segmente** miteinander **verbinden**. Auch hier ist zu beachten, dass die räumliche Ausdehnung zwischen zwei Endstationen begrenzt ist. ([9], S.50)

### 3.3.2 Sternkoppler

Sternkoppler sind funktionsmäßig betrachtet Multiport Repeater, d.h. sie **können** wie auch der Multiport Repeater **mehrere Netzsegmente** miteinander **verbinden**. Sie können jedoch weitaus mehr Segmente konzentrieren und darüber hinaus **verschiedene Medien** wie KOAX oder Glasfaser **integrieren**. ([9], S.50)

Bei Ethernet-Netzwerken, die mit Twisted Pair-Verkabelung aufgebaut werden, ist an jedem Port des Sternkopplers (in diesem Fall meist **Hub** genannt) nur eine Station (oder ein weiterer Hub) angeschlossen. Der Hub übernimmt dann die Weiterleitung der auf einem Port ankommenden Pakete an alle anderen Ports.

## 3.4 Bridge

Bridges sind Netzkoppeleinheiten, die verschiedene Segmente eines Netzes zu einem Gesamtnetz verbinden. Eine Bridge kann zur Erweiterung der räumlichen Ausdehnung eines Netzes verschiedene Netzsegmente auch dann noch miteinander verbinden, wenn aufgrund einer beschränkten erlaubten Anzahl kein Repeater mehr eingesetzt werden darf. Dies rührt daher, dass bei einer Kopplung mittels **Bridge eine logische Trennung der beiden Netzsegmente** vorgenommen wird.

D.h. Bridges übernehmen die **Zwischenspeicherung** der Datenpakete und sorgen für einen **kollisionsfreien Weitertransport**. Man spricht hier auch davon, dass eine Bridge das Netzwerk in mehrere Kollisionsdomänen (Collision Domains) aufteilt, da die Auswirkungen einer Kollision auf ein Segment begrenzt bleibt.

**Im Fehlerfall haben also Fehler durch die logische Aufteilung in Teilnetze nur noch eine begrenzte Wirkung**, was beim Einsatz von Repeatern nicht der Fall ist. Hier wird im Fehlerfall das komplette Netzwerk beeinträchtigt. ([5], S.56-57)

Aufgrund der weiter unten geschilderten Probleme finden meist **Bridges** Verwendung, die **auf Ebene 2a** (Medium Access Control MAC) des OSI-Modells **arbeiten**, so dass mittels einer solchen Bridge eine Kopplung **nur zwischen Netzen gleichen Typs** möglich ist. Grundsätzlich sind Bridges in der Lage, sowohl gleichartige, als auch, bezüglich ihrer physikalischen Eigenschaften und des Zugangsverfahrens, unterschiedliche Netzwerke miteinander zu verbinden. Bei der Verwendung solcher Mixed Media Bridges treten jedoch einige Probleme auf, die im folgenden kurz angesprochen werden. ([1], S.14-15; [2], Kap. 20.2; [12], Kap. 3)

Bei den insgesamt 9 möglichen Kombinationen der **Kopplung zweier von IEEE** (IEEE: Institute of Electrical and Electronic Engineers) **standardisierten Netze**, wie z.B. IEEE 802.3 (Ethernet) mit IEEE 802.5 (Token Ring) mit einer Bridge entstehen die folgenden Probleme: ([2], S.73-74, [13], S.325-328)

- Verwendung unterschiedlicher Rahmenformate (benötigt Anpassung, Berechnung einer neuen Prüfsumme)
- unterschiedliche Paketgrößen: Die Paketgrößen variieren zwischen 1518 Bytes (802.3) und 8191 Bytes (802.4). Damit müssten zu große Pakete segmentiert und wieder zusammengesetzt werden, dies ist jedoch bei einem Koppelelement auf Schicht 2 nicht möglich.
- Prioritäten: Token Bus enthält Prioritäten für Pakete - diese gehen verloren
- Acknowledgement (Im Token Ring erhält der Sender vom Empfänger zur Bestätigung des Empfangs eine Kopie des abgesendeten Paketes): Token Ring liefert Acknowledgement für die Ablieferung an der Destination. Unklar ist, wie die Bridge dies umsetzen soll.
- unterschiedliche Übertragungsraten: Bei der Übertragung von einem schnelleren in ein langsameres Netz muss die Bridge eventuelle Pakete zwischenspeichern. Dies ist kein grundsätzlich neues Problem für Bridges; es erhöht nur die Wahrscheinlichkeit eines Paketverlustes.

**Mixed Media Bridges** arbeiten auf Ebene 2b (Logical Link Control LLC) des OSI Modells und können somit auch Netzwerke mit unterschiedlichem MAC-Protokollen koppeln (z.B. Ethernet und Token Ring). Da bei einer solchen Umsetzung aber viele Probleme auftreten (siehe oben) und die Art und Weise, wie die Umsetzung zwischen unterschiedlichen Netzwerktypen erfolgen soll, nicht standardisiert ist, gibt es auch keine „richtige“ Art der Umsetzung. ([12], Kap. 31)

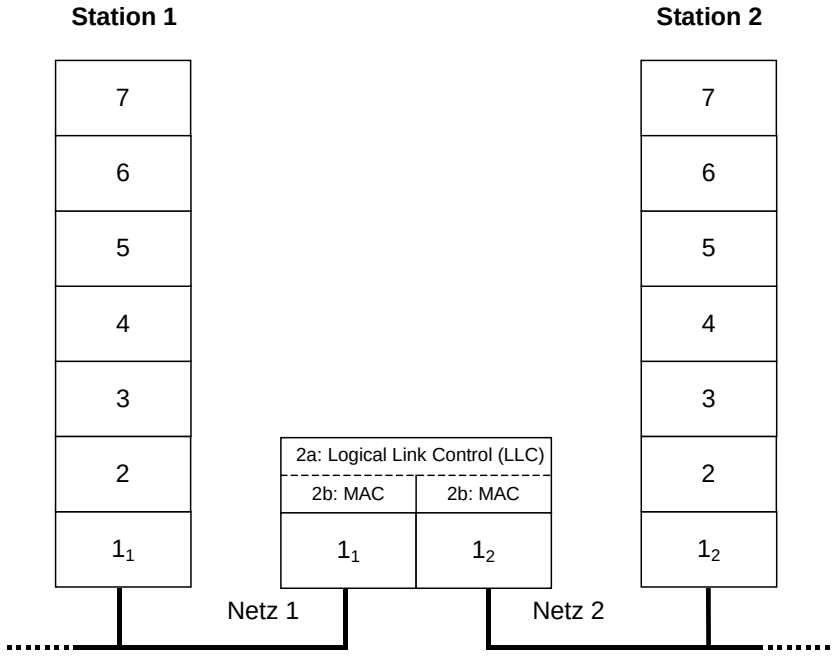



*Bild 3-9 Netzkopplung über eine (Mixed Media-)Bridge*

Die in den Datenpaketen enthaltenen Informationen werden, außer den MAC-Layer-Source- und Destination-Adressen (MAC: Media Access Control), von einer Bridge nicht ausgewertet. Daher **können Bridges mit jedem Kommunikationsprotokoll der Schicht 3 zusammenarbeiten,** sind also protokollunabhängig. Deshalb ist es vollkommen gleichgültig ob es sich bei den übertragenen Datenpaketen um OSI- oder TCP/IP-Pakete handelt. Darin liegt der große Vorteil von Bridges. ([3], Kap. 20.2)

Abhängig vom Hersteller unterstützen Bridges unterschiedliche Routing-Mechanismen. IBM ist derzeit der einzige Hersteller, der Source-Routing verwendet, alle anderen Hersteller verwenden Transparente Bridges mit dem Spanning-Tree Verfahren.

Es sei hier vermerkt, dass **Bridges** im Gegensatz zu den unten beschriebenen Routern und Gateways **nicht in der Lage** sind, **alternative Wegewahl** zu unterstützen, da sie nicht das gesamte Netzwerk kennen.

### 3.4.1 Filternde Bridge

Um Datenverkehr in Teilnetzen, deren Stationen häufig miteinander kommunizieren, vom Gesamtnetz zu isolieren, werden filternde Bridges eingesetzt. Eine **filternde Bridge** ist ein **intelligentes System** im Netzwerk, welches nur die Pakete von einem Teilnetz zum anderen übergibt, die von einer Station des einen Teilnetz für eine Station in einem anderen Teilnetz bestimmt sind. Dadurch wird das übrige Netz nur dann belastet, wenn Daten erzeugt werden, die für andere Stationen in einem anderen Teilnetz bestimmt sind. Der **interne Datenverkehr** der einzelnen Teilnetze **wird** also **lokal begrenzt**, lediglich der zu anderen Teilnetzen übergreifende Verkehr wird über die Bridge weitergeleitet. ([1], S.14-15; [2], Kap. 20.2)

Auch in punkto **Sicherheit** bietet die Technik des Filtering einige Vorteile. Durch das logische Trennen der Netze wird durch die filternde Bridge ein Weg geschaffen, den **übergreifenden Verkehr** zu **kontrollieren**. Datenpakete können gezielt vom Zugang ins übrige Netz ausgeschlossen werden. Filtertabellen, die vom Netzadministrator festgelegt werden, erlauben unter anderem, die Quelladresse des Senders und die Zieladresse zu überprüfen. Nur dem Netzknoten, dem die entsprechende Zugangsberechtigung erteilt wurde, wird der Zugriff auf andere Netzwerkbereiche erlaubt.

Für das Filtering sind unterschiedliche Filter möglich. Die meisten Bridges werten die im Datenpaket enthaltenen Empfänger- oder Absenderadressen aus. Ein Beispiel hierfür ist der sogenannte Source-Address-Filter. **Source-Address-Filter** dienen dazu, Datenpakete bestimmter Stationen nicht in andere Teilnetze zu übertragen. Dies kann beispielsweise bei Messwertaufnehmern, die periodisch große Datenmengen generieren, sinnvoll sein. ([3], Kap. 20.4.3)

Manche Bridges verwenden zusätzlich einen sogenannten Protocol-Type-Filter. **Protocol-Type-Filter** werden eingesetzt, um Datenpakete eines bestimmten Protokolls nicht in andere Teilnetze zu übertragen. Eine Variante dieses Filters gibt bestimmten Protokollen bei der Übertragung eine höhere Priorität.

Die Informationen, die eine Bridge für das Filtering benötigt (z.B. die verschiedenen Adressen der Stationen der zu koppelnden Netze) werden in einer Adress-Tabelle gehalten. Hierbei gibt es zwei Realisierungsmöglichkeiten:

Bei der ersten Möglichkeit entscheidet die Bridge anhand einer **statischen Tabelle**, in der die Stationsadressen des gekoppelten Netz abgelegt sind, ob das Paket weitergegeben werden muss. Man spricht hier von einer **statischen Brücke** oder einer **einfachen Brücke (Simple Bridge)**. ([9], S.78)

Bei der zweiten Möglichkeit wird ebenfalls anhand einer solchen **Tabelle** entschieden, allerdings ist diese **dynamisch**. Es handelt sich dann um eine sogenannte **self-learning Bridge** oder auch **transparente Bridge**, welche die Adressen auf jeder Netzseite selbständig erfasst und für die Entscheidung zur Weitergabe verwendet (Einzelheiten im Abschnitt Transparente Bridge).

Um die heute üblichen filternden Bridges am Markt gezielt vergleichen zu können, sind einige **Leistungs-Parameter** zu bewerten. Hierzu wird die sogenannte **'Filter-Rate'** und die **'Forward-Rate'** herangezogen. Die Filterrate kennzeichnet, wie viele Datenpakete pro Zeiteinheit von der Bridge auf 'zu transportieren' oder 'zu blockieren' kontrolliert werden können. Ein realistischer Vergleich ist nur dann möglich, wenn die

zugrunde gelegten Paketgrößen bekannt sind. Seriöse Angaben beziehen sich auf die kleinste Paketgröße von beispielsweise 64 Byte im Ethernet und 23 Byte im Token Ring. Die Filterrate wird natürlich von der Anzahl der Tabelleneinträge beeinflusst.

Mit der **Forward Rate** wird die Zahl der maximale transportierbaren Rahmen gekennzeichnet. Die Filter Rate wie auch die Forward Rate sollten möglichst nahe an der theoretischen, maximalen Leistung des LANs liegen. Nur dann ist gewährleistet, dass auch bei kurzzeitiger Hochbelastung kein Datenverlust entsteht. ([5], S.60)

### 3.4.2 Nichtfilternde Bridge

Nichtfilternde Bridges, welche den oben beschriebenen Filter-Mechanismus nicht besitzen und somit sämtliche Datenpakete weiterleiten, sind heutzutage nur noch selten anzutreffen. Sie wurden bis zum Aufkommen der filternden Bridges zur Kopplung von Netzsegmenten eingesetzt, um die Ausdehnung der Netze auszuweiten, was bei Repeatern aufgrund der oben beschriebenen Eigenschaften ja nur begrenzt möglich ist. **Nichtfilternde Bridges leiten** also genau wie ein Repeater **sämtlichen Datenverkehr weiter**, sie **übernehmen** aber im Unterschied zum Repeater die **Zwischenspeicherung der Datenpakete** und **sorgen für** einen **kollisionsfreien Weitertransport**. Heute werden allerdings fast ausschließlich filternde Bridges eingesetzt, da sie die oben aufgeführten Vorteile bieten.

Zur Beurteilung nichtfilternder Bridges kann nur die Forward-Rate herangezogen werden.

### 3.4.3 Transparente Bridge

Wesentliches Merkmal einer transparenten oder selbstlernenden Bridge ist, das diese Bridge **in ein bestehendes Netzwerk integriert werden kann und ohne weitere Konfiguration ihre Aufgabe erfüllt**. Die Bridge ist völlig transparent, das heißt, es ist nach dem Einbau nicht nötig, die Netzwerkhardware zu ändern, Software zu verändern, die Adressierung zu ändern oder Routing-Tabellen oder Parameter zu laden. Transparente Bridges müssen sich ihre Kenntnis über das Netzwerk im laufenden Betrieb selbst erwerben. Sie arbeiten daher im Gemischtmodus (Promiscuous Mode) und akzeptieren jeden Rahmen von allen angeschlossenen LANs. Anhand der MAC-Zieladresse des eingehenden Rahmens kann eine Bridge entscheiden, ob sie den Rahmen weiterleitet oder nicht.

In Bild 3-10 ist Bridge 1 an LAN 1 und LAN 2 angeschlossen und Bridge 2 an LAN 2, LAN 3 und LAN 4. Empfängt nun Bridge 1 auf LAN 1 einen Rahmen, der an A adressiert ist, kann dieser verworfen werden, weil sich dieser bereits im richtigen LAN befindet. Ein Rahmen, der auf LAN 1 für C, D oder E eingeht, muss jedoch weitergeleitet werden.

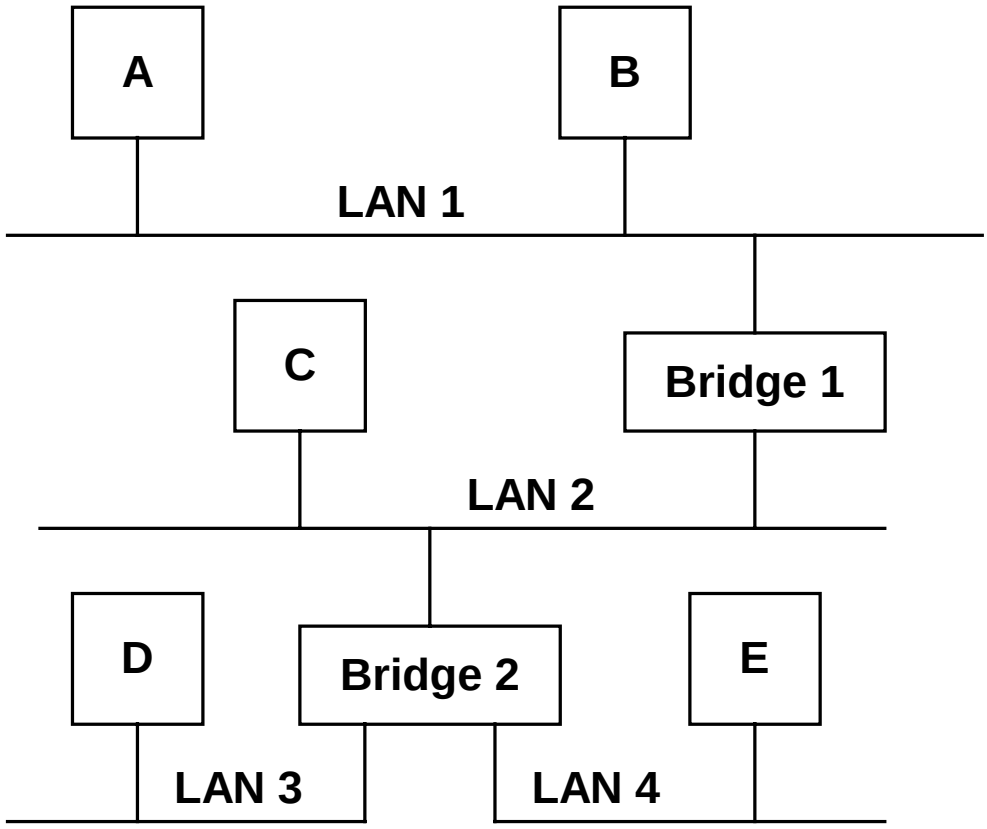



*Bild 3-10 Kopplung mehrerer LANs mit zwei Bridges*
*Rahmenweiterleitung*

Kommt ein Rahmen an, muss die Bridge entscheiden, ob der Rahmen verworfen oder weitergeleitet wird und wenn er weitergeleitet wird, auf welches LAN er ausgegeben werden soll. Diese Entscheidung wird aufgrund der in der Bridge hinterlegten Hash-Tabelle getroffen. In dieser Tabelle ist hinterlegt, welcher Empfänger auf welcher Ausgangsleitung zu erreichen ist. In der Tabelle von Bridge 1 zum Beispiel wäre hinterlegt, dass die Stationen C, D und E über LAN 2 zu erreichen sind. Das D und E nicht direkt an LAN 2 angeschlossen sind, ist dabei nicht von Bedeutung.

Wird eine Bridge an ein LAN angeschlossen, ist die Hash-Tabelle natürlich leer. Diese wird gefüllt, indem die MAC-Adressen der eingehenden Rahmen in der Tabelle hinterlegt werden. Empfängt Bridge 1 beispielsweise einen Rahmen von A auf LAN 1, wird in der Tabelle hinterlegt, dass A über LAN 1 adressiert wird. Jeder nachfolgend an A adressierte Rahmen wir dann an LAN 1 weitergegeben, Rahmen, die auf LAN 1 für A eingehen, werden verworfen. Dieser Algorithmus wird **Backward Learning** genannt. Die Einträge werden ständig im laufenden Betrieb überprüft und alte Einträge werden periodisch aus den Tabellen gelöscht. Damit ist ein Rechner, der in ein anderes LAN umzieht, schon nach kurzer Zeit wieder betriebsbereit.

Solange für das adressierte Ziel kein Eintrag in der Tabelle besteht, wird der Rahmen auf allen angeschlossenen LANs ausgegeben, mit Ausnahme des LANs, von dem der Rahmen kommt - diesen Vorgang nennt man **Flooding**.

Zusammenfassend lässt sich also sagen:

1. Sind Quell- und Ziel-LAN identisch, wird der Rahmen verworfen
2. Sind Quell- und Ziel-LAN verschieden, wird der Rahmen weitergeleitet
3. besteht kein Eintrag für das Ziel-LAN, wird die Flooding-Technik angewandt.

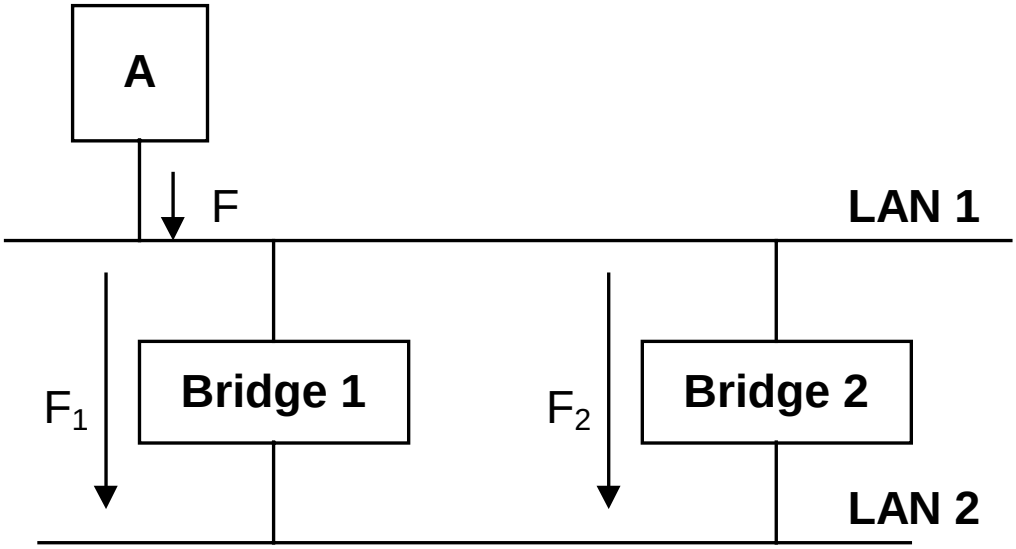



*Bild 3-11 Zwei parallele transparente Bridges*
*Schleifenbildung*

Probleme entstehen, wenn, wie in obiger Abbildung, zur Erhöhung der Zuverlässigkeit zwei oder mehr Bridges zur Verbindung zweier LANs eingesetzt werden. Sendet Station A einen Rahmen an ein unbekanntes Ziel, folgen die Bridges dem oben beschriebenen Algorithmus und kopieren den Rahmen auf LAN 2. Kurz danach empfängt Bridge 1 den von Bridge 2 auf LAN 2 kopierten Rahmen F2 mit unbekanntem Ziel und kopiert diesen wiederum auf LAN 1, wo dieser wieder von Bridge 2 empfangen und auf LAN 2 kopiert wird. Entsprechendes geschieht mit Rahmen F1. Es entsteht dadurch ein unendlicher Kreislauf. ( [13], S.329-330)

### 3.4.4 Spanning Tree

In großen Netzwerken gibt es in der Regel **redundante Wege**, die besonders dann von Bedeutung sind, wenn ein Weg z.B. durch eine defekte Netzkoppeleinheit ausfällt. Der **Nachteil** von redundanten Wegen ist die Gefahr der im vorigen Abschnitt beschriebenen **kreisenden Daten (Loops)**, die das Netzwerk belegen und den Durchsatz dadurch reduzieren. Um solche Closed Loops zu vermeiden, wurde der '**Spanning-Tree**' Algorithmus entworfen, welcher von IEEE im Standard 802.1 standardisiert wurde. ([3], Kap. 20.4.1)

**Beim Spanning-Tree Verfahren kommunizieren die Bridges untereinander und die aktuelle Topologie wird mit einem überspannenden Baum (Spanning Tree) überlagert.** Das Spanning Tree Verfahren ist ein sogenanntes **Teiltopologie-Verfahren**. Bei einem Teiltopologie-Verfahren wird nur ein Teil der Netztopologie verwaltet. Bei Spanning Tree wird darüber hinaus auch nur ein Teil der Netztopologie benutzt, was zwar zur Folge hat, dass Netzressourcen verschwendet werden, dafür aber keine geschlossenen Schleifen entstehen können. Die genutzte Teiltopologie, die sich durch das Deaktivieren von redundanten Verbindungswegen ergibt, ist eine Baum-Struktur aktiver Verbindungen. Alle vorhandenen redundanten Wege werden nach bestimmten Kriterien, wie Leitungskosten oder Leitungskapazität, in einen Backup-Status versetzt, so dass alle vernetzten Punkte durch genau einen Weg miteinander verbunden sind.

Daraus ergeben sich die folgenden Eigenschaften:

- alle vernetzten Punkte sind von allen anderen vernetzten Punkten aus erreichbar
- es gibt zwischen zwei beliebigen vernetzten Punkten genau einen definierten Weg
- es gibt zwischen zwei beliebig vernetzten Punkten keine Loops

Fällt zwischen zwei Knoten eine Verbindung, d.h. eine Strecke aus, wird automatisch die im Backup-Status befindliche Strecke aktiviert, sie befinden sich also im sogenannten **hot Standby**. Zum Aufbau eines Spanning-Tree wird folgendes - hier stark vereinfachtes - Verfahren eingesetzt. ([2], S.86-87, [3], Kap. 20.4.1; [9], S.93-97, [13], S.332)

Zunächst wird die sogenannte Wurzel (Root) bestimmt. Die Wurzel ist die Station mit der niedrigsten Identifikations-Nummer. Dazu senden alle Knoten ihre Identifikations-Nummer aus, (eine vom Hersteller installierte und garantiert eindeutige Seriennummer) einschließlich ihren gegenwärtig angenommenen **Pfadkosten** zur Wurzel (werden zunächst mit 0 initialisiert).

Der Begriff Pfadkosten ist wie folgt definiert:

Pfadkosten = 1000/Leitungskapazität in Mbit/s

Daraus ergeben sich z.B. für Ethernet (10 Mbit/s) Pfadkosten von 100, für Token Ring mit 4 Mbit/s Pfadkosten von 250. ([9], S.95)

Sieht ein Knoten ein Paket eines anderen Knoten mit niedrigerer Identifikations-Nummer, so akzeptiert er diesen als Wurzel und sendet in der Folge seine Entfernung und die Identifikations-Nummer der angenommenen Wurzel aus. Empfängt er ein Paket eines Knoten mit geringerer Entfernung zu seiner

angenommenen Wurzel, so ersetzt er dessen Angabe der Pfadkosten bei sich. Nach einigen Iterationen sind allen Knoten im Netz, die Wurzel und ihre jeweilige kürzeste Entfernung (Weg mit geringsten Pfadkosten) zur Wurzel bekannt.

In einem zweiten Schritt wird der sogenannte Vorgänger sowie der Nachfolger bestimmt. Jeder Knoten sucht sich seinen Vorgänger auf dem kürzesten Weg zur Wurzel. Kennt er mehrere Knoten mit gleich kurzer Entfernung zur Wurzel, so wird der Knoten mit der kleinsten Identifikations-Nummer ausgewählt.

Jeder Knoten, der sich einen anderen Knoten als Vorgänger ausgewählt hat, teilt ihm das mit, so dass ein Knoten auch seinen Nachfolger kennt.

Jede Netzkoppeleinheit bringt durch den oben beschriebenen Mechanismus also folgende drei Dinge in Erfahrung:

- welche Netzkoppeleinheit ist die Wurzel
- wie weit ist die Wurzel entfernt
- welches ist die Netzkoppeleinheit, die mich zur Wurzel führt (Vorgänger)

Wenn der Spanning-Tree aufgebaut ist, kann ein Knoten jedes Paket auf allen aktiven (d.h. nicht im Backup-Status befindlichen) Ästen außer dem Empfangs-Ast weitersenden; dies führt dann zu keinen Loops.

Zur Überwachung des Spanning-Tree werden periodisch von der Wurzel des Baumes bis zu den Blättern hinunter Überwachungsnachrichten zu allen Nachfolgern gesendet. Wird dabei festgestellt, dass nicht mehr alle Nachfolger erreichbar sind, so läuft obiges Verfahren im Regelfall erneut an. Für diesen Zeitraum ruht anderer Verkehr. ([2], S.86-87)

Das folgende Bild zeigt ein einfaches Beispiel für einen Spanning-Tree:

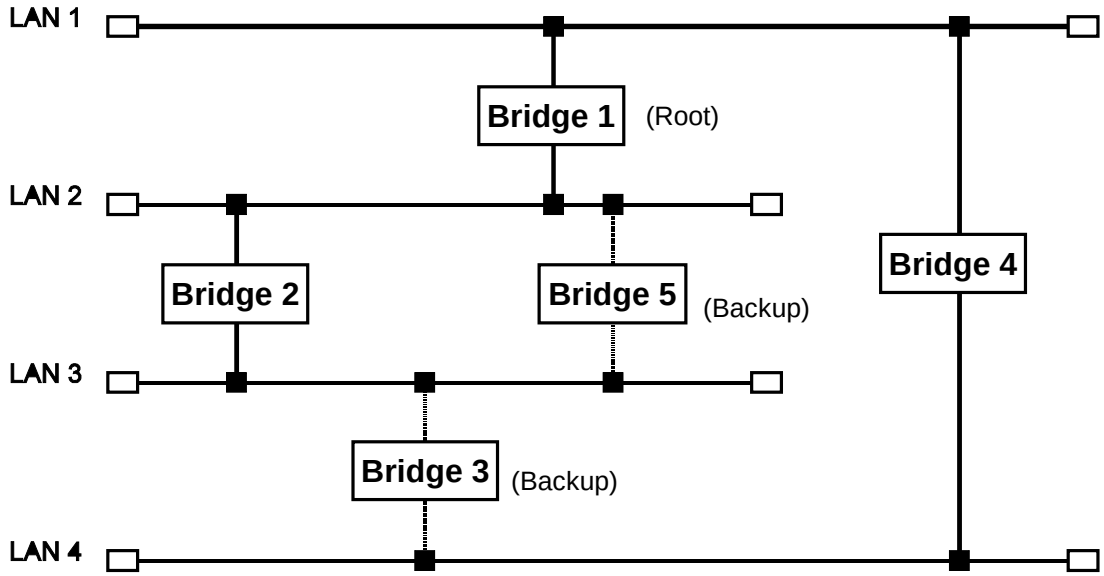



*Bild 3-12 Beispiel für Spanning Tree*

### 3.4.5 Source Route Bridging

Das Source Routing Verfahren wird fast ausschließlich in Token Ring Netzwerken eingesetzt und geht davon aus, dass die sendende Station weiß, ob der Empfänger sich im selben LAN befindet oder nicht. Soll ein Rahmen in ein anderes LAN übertragen werden, setzt die Quellmaschine das High-Order-Bit der Zieladresse auf 1 und trägt in den Rahmen-Header den genauen Pfad zum Ziel ein. Der Pfad setzt sich dabei folgendermaßen zusammen: Jedes LAN hat eine eindeutige, aus 12 Bit bestehende Nummer und jede Bridge hat eine aus 4 Bit bestehende Nummer, die sie innerhalb des LANs eindeutig identifiziert. Ein Pfad besteht aus einer Abfolge solcher LAN - Bridge Nummern. Erkennt eine Bridge einen Rahmen mit gesetztem High-Order-Bit, überprüft sie die Pfadinformationen und sucht die Nummer des LANs, aus dem der Rahmen gekommen ist. Folgt auf die LAN-Nummer die eigene Nummer, kopiert sie das Paket in das angegebene LAN.

Dieses Verfahren hat bezüglich der Wegesuche sehr einfache Netzkoppeleinheiten ohne Tabelle zur Folge, allerdings muss nun jede Station zu jeder Zeit genau die Konfiguration des Netzes kennen und den Weg bereits beim Senden der PDU auswählen. **Ist der Pfad zum Ziel einer Station nicht bekannt, sendet sie einen Broadcast-Rahmen** aus, der von allen Bridges weitergeleitet wird. **Auf dem Weg zurück zum Sender fügen die Bridges ihre Kennung in den Suchrahmen (Discovery Frame) ein**. Damit kann der Sender den zurückgelegten Weg sehen und die optimale Route ermitteln. Mit diesem Verfahren wird zwar zweifellos die beste Route ermittelt, jedoch hat dies eine große Anzahl von Suchrahmen zur Folge. Wird zum Beispiel jedes LAN über 3 Brücken mit dem nachfolgenden LAN verbunden, erzeugt ein in LAN 1 abgesendeter Suchrahmen in LAN 2 drei Suchrahmen, was in LAN 3 dann zu neun Suchrahmen führt usw. , es sind also allgemein $3^{N-1}$ Rahmen im Umlauf. Bei einer Konfiguration mit einem Dutzend solcher Bridge-Gruppen kreisen im letzten LAN dann über eine halbe Million Suchrahmen. Bei der transparenten Bridge läuft zwar mit dem Flooding-Verfahren ein ähnlicher Mechanismus ab, jedoch wirkt sich dies nur entlang des Spanning Tree aus, so dass die Zahl der übertragenen Rahmen linear und nicht exponentiell mit der Netzgröße steigt.

Im Gegensatz zu einer Transparenten Bridge ist beim Source Routing eine Änderung der auf den Stationen laufenden Software nötig, da alle Stationen das Bridge-Schema kennen und daran teilnehmen müssen. Ein weiterer Unterschied besteht darin, das transparente Bridges Störungen, zum Beispiel durch Ausfall einer Bridge, durch die Kommunikation untereinander erkennen und den Ausfall selbständig beheben können (falls ein alternativer Pfad besteht). Während sich transparente Bridges selbst konfigurieren, müssen die LAN- und Bridge-Nummern manuell vergeben werden, was eventuell zu schwer lokalisierbaren Fehlern führen kann (z.B. doppelte LAN-Nummern, die zu kreisenden Rahmen führen). ([1], S.10; [2], S.77-78; [6], S.39; [13], S.332-335)

### 3.4.6 Multiport Bridge

Nachdem lange Zeit **2-Port-Bridges** den Markt dominierten, die **genau zwei Teilnetze** miteinander **verbinden**, werden heute aufgrund des Bedarfs vieler Teilnetze immer häufiger die sogenannten **Multiport-Bridges** eingesetzt, die **mehrere Teilnetze** miteinander **verbinden** können. Sie verbinden in der Regel vier oder acht Teilnetze. Multiport Bridges erleichtern den Aufbau von Baum-Strukturen (Mehrfachverzweigungen), außerdem wird durch die Konzentration mehrerer Ports in einer Bridge die **Anzahl notwendiger Brücken reduziert** und die **Lastverteilung auf verschiedene Verbindungen** erleichtert. ([9], S.61-62)

### 3.4.7 Switches

Switches stellen eine Sonderform von Multiport-Bridges dar, die vor allem in Ethernet Netzwerken mit Twisted Pair Verkabelung von großer Bedeutung sind. Wie die folgende Abbildung zeigt, kann ein Switch gleichzeitig mehrere interne Verbindungen **mit voller Geschwindigkeit** bereitstellen.

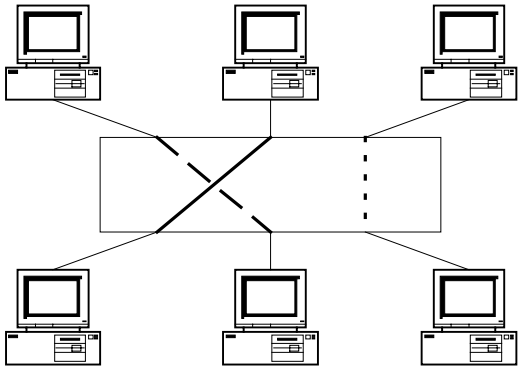


*Bild 3-13 Prinzipielle Wirkungsweise eines Switches*

Die Anzahl der gleichzeitigen Verbindungen entspricht üblicherweise der halben Portanzahl. So wären zum Beispiel bei einem Switch mit 8 Ports vier gleichzeitige Verbindungen zwischen jeweils 2 Station mit maximaler Geschwindigkeit möglich. Der Switch ordnet dabei anhand der MAC-Zieladresse dem ankommenden Rahmen den passenden Ausgangsport zu.

Ein wichtiger Begriff im Zusammenhang mit Switches ist die aggregierte Bandbreite. Damit wird die maximale Übertragungsrate, die der Switch zur Verfügung stellen kann, bezeichnet. Diese Größe ist abhängig von der Anzahl der Ports und der Übertragungsgeschwindigkeit des Netzes:

$$\text{aggregierte Bandbreite} = \frac{\text{Portanzahl}}{2} * \text{Übertragungsgeschwindigkeit}$$

Die interne Verbindung der Ports, die sogenannte Backplane, muss natürlich in der Lage sein, die hohen Übertragungsgeschwindigkeiten innerhalb des Gerätes zu bewältigen. Unterstützt der Switch Vollduplex-Verbindungen (gleichzeitiges Senden und Empfangen, nur bei direktem Anschluss eines Rechners an einen Port möglich), muss die oben angegebene Formel um eine Multiplikation mit dem Faktor 2 ergänzt werden.

Verwendet man einen 12-Port Switch mit 10 Mbit/s als Collapsed Backbone, ergibt sich nach obiger Formel eine aggregierte Bandbreite von 60 Mbit/s, im Gegensatz zu 10 Mbit/s bei einem Distributed Backbone. Obwohl der berechnete Maximalwert in der Praxis wohl in den seltensten Fällen erreicht werden wird, bewirkt ein Switch im Gegensatz zu einem Hub oder einem Distributed Backbone auf jeden Fall eine deutliche Steigerung der Transferrate.

An einen Port kann entweder eine Arbeitsstation direkt angeschlossen werden oder es werden mehrere Arbeitsstationen über einen Hub mit dem Port verbunden. Die Art des Anschlusses hängt vom Bandbreitenbedarf der angeschlossenen Stationen ab. Wird ein Rechner direkt an einen Port angeschlossen, kann die Verbindung als Vollduplex-Verbindung betrieben werden, falls die Netzadapterkarte des Rechners und der Switch dies unterstützen. Vollduplex bedeutet, das der Rechner gleichzeitig Daten senden und empfangen kann, was die zur Verfügung stehende Bandbreite verdoppelt.

Switche, bei denen alle Ports mit derselben Übertragungsrate arbeiten, sind vor allem für Peer-to-Peer-Netze geeignet, bei denen alle Geräte auf die Dienste der anderen Geräte zugreifen. Da aber in den meisten Netzen eine Serverbasierte Struktur verwendet wird und dabei der größte Teil des Verkehrs zwischen einem Server und den Arbeitsstationen stattfindet, werden Switches oftmals mit einem Uplink-Port angeboten. Dieser bietet dann beispielsweise eine 100 MBit-Verbindung zum Server und mehrere 10 MBit-Verbindungen zu den Arbeitsstationen.

Bild 3-14 zeigt ein Beispiel, bei dem die Desktop-PCs sich einen 10Mbit-Port teilen. Die sogenannten „Power-User“, die einen höheren Bedarf an Bandbreite haben, sind direkt an einen Port des Switches angeschlossen. Der Server ist mit einer 100Mbit Verbindung an den Switch angeschlossen.

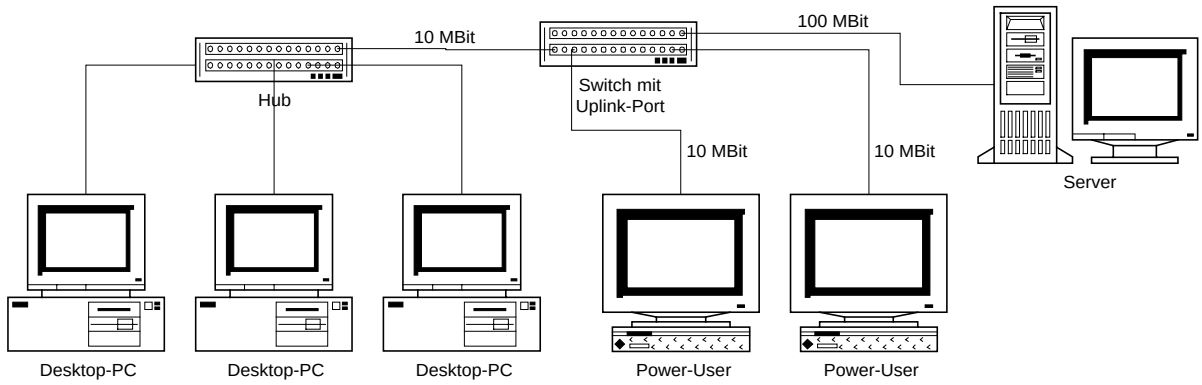



*Bild 3-14 Beispiel für Einsatz eines Switch*

Für das Switching bestehen derzeit zwei unterschiedliche Strategien: das sicherere Store-and-Forward und das schnellere Cut-Through-Verfahren.

#### 3.4.7.1 Store-and-Forward-Switching

Bei diesem Verfahren wird zunächst, wie bei einer normalen Bridge, das ankommende Datenpaket komplett gespeichert und überprüft. Ist das Paket, zum Beispiel aufgrund einer Kollision, beschädigt, wird es verworfen. Dadurch werden nur korrekte Pakete weitergeleitet. Durch die Zwischenspeicherung der Daten kommt es jedoch zu einer erheblichen Verzögerung der Datenübertragung. Die Latency-Zeiten (Lese- und Verarbeitungszeiten) liegen je nach Paketgröße bei 100 bis 1300µs. Ein Switch, der mehrere langsamere Ports auf einen schnelleren Port umsetzt, arbeitet immer als Store-and-Forward-Switch, da hier die ankommenden 10MBit Verbindungen in einen geordneten 100MBit-Datenstrom umgesetzt werden müssen.

#### 3.4.7.2 Cut-Through-Switching

Die Weiterleitung der Datenpakete beginnt beim Cut-Through-Switching sobald die 6 Byte lange Zieladresse gelesen wurde. Dadurch kann der Switch nach den ersten 20 bis 30 Byte die Verbindung zwischen dem Eingangs- und Ausgangsport herstellen. Die Verzögerungszeit pro Datenpaket liegt dadurch nur bei etwa 30 bis 60µs, es werden jedoch auch eventuell beschädigte Pakete weitergeleitet.

#### 3.4.7.3 Hybrid-Switching

Das Hybrid-Switching-Verfahren, auch adaptives Switching genannt, versucht die Vorteile von Store-and-Forward und Cut-Through-Switching zu vereinen. Ein Hybrid-Switch arbeitet normalerweise als Cut-Through-Switch, beobachtet dabei aber ständig die Anzahl der beschädigten Pakete. Übersteigt diese Anzahl einen Grenzwert, schaltet der Switch auf Store-and-Forward Betriebsart um und verhindert damit, dass beschädigte Pakete in die anderen Netzsegmente weitergeleitet werden. Geht die Fehlerrate wieder zurück, arbeitet er wieder mit der schnelleren Cut-Through-Betriebsart.

## 3.5 Router

Ein Router koppelt Netze mit unterschiedlichen Netzadressen auf der Vermittlungsschicht (Schicht 3), dies bedeutet, dass die zu koppelnden Netze in den Schichten 1 und 2 unterschiedlich sein dürfen.

Router können protokollabhängig sein und werden dann als **Single Protocol Router** bezeichnet (TCP/IP-Router, DECnet-Router), es gibt aber auch sogenannte **Multiple Protocol Router**, die mehrere Protokolle verstehen und transportieren können.

Ein Router ist nicht transparent, d.h. der Absender der Datenpakete muss von der Existenz des Routers wissen und diesen direkt adressieren, damit die Datenpakete weitergeleitet werden können.

Dies macht die Arbeitsweise eines Routers im Vergleich zu einer Bridge wesentlich effizienter, da er nur die Datenpakete aufnehmen muss, die für ihn bestimmt sind. Aus der eben beschriebenen Eigenschaft, dass Router nicht transparent sind, resultiert natürlich, dass ein **Router keine Filterfunktionen** besitzt, da er sowieso nur Datenpakete erhält, die an eine Station in einem anderen Netzsegment weitergeleitet werden sollen.

Die Vermittlungsschicht kann in drei Teilschichten aufgeteilt werden, wobei die Teilschicht 3a noch teilnetzspezifisch ist. Die Anpassung des Protokolls der Teilschicht 3a an das Protokoll der Teilschicht 3c wird durch das Protokoll der Teilschicht 3b vorgenommen. Siehe hierzu Bild 3-15. ([1], S.16)

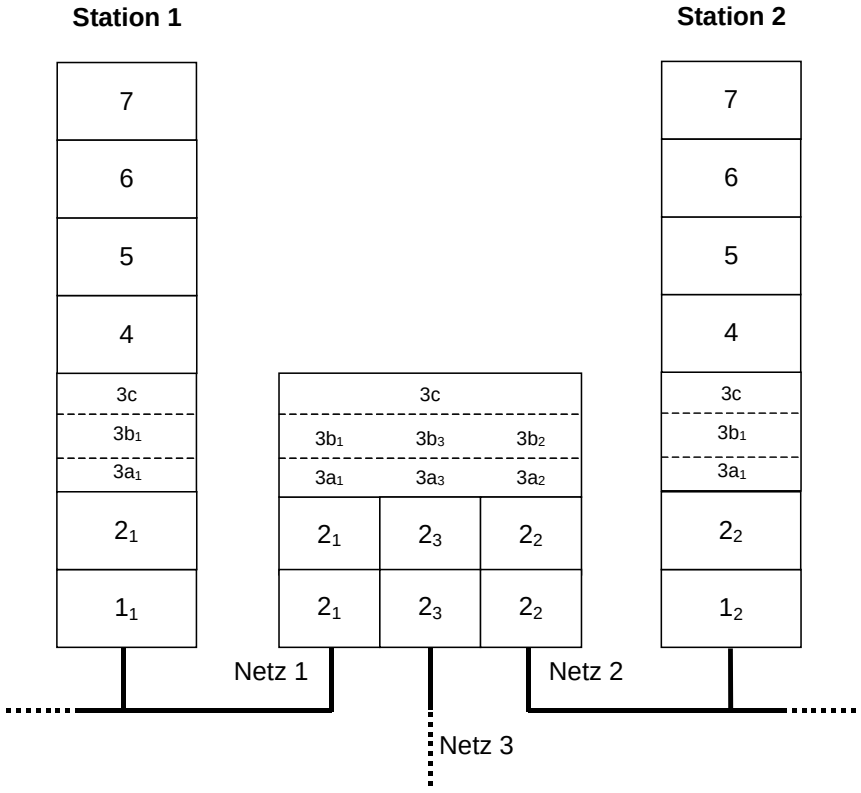



*Bild 3-15 Netzkopplung über einen Router (Bsp. für drei zu koppelnde Netze)*

Da Router in der Lage sind, sich gegenseitig Informationen zu übermitteln und mit Endgeräten zu kommunizieren, können Router mit anderen Routern oder Hosts die Routing-Informationen austauschen und aktualisieren. Ein **Router** ist deshalb **in der Lage**, genau **zu wissen, wo eine Adresse im Netzwerk zu finden ist**. Als Protokoll wird meist ein globales Internetprotokoll verwendet, das insbesondere die Wegesuche im globalen Netz vornimmt.

Ein Router arbeitet nicht mit der MAC Source- und Destination-Adresse, sondern direkt mit der IP-, DECnet- usw. Adresse. Mit Hilfe dieser Adressen und den dazugehörigen Routing-Protokollen baut sich ein Router die dazugehörigen **Routing-Tabellen** auf. Bei den meisten Routern geschieht dieser Aufbau automatisch nach dem Einschalten des Routers. In Ausnahmefällen, wenn z.B. genau feststeht, dass Routen sich nicht ändern, können auch Router mit statischen Routing-Tabellen verwendet werden. ([1], S.15-16; [2], Kap. 20.3; [4], S.52; [7], S.16; [9], S.184-185)

Grundsätzlich unterscheidet man zwei Arten von Routing-Protokollen:

- ein *Interior Gateway Protocol (IGP)* wird innerhalb eines Netzes eingesetzt. Das bekannteste IGP dürfte wohl das *Router Information Protocol (RIP)* sein, das allerdings aufgrund seiner Einschränkungen und Probleme in letzter Zeit durch *Open Shortest Path First (OSPF)* abgelöst wird
- zur Kommunikation zwischen Routern in autonomen Netzen wird ein *Exterior Gateway Protocol (EGP)* eingesetzt. Das seit 1984 bestehende Protokoll EGP wurde mittlerweile weitgehend durch das *Border Gateway Protocol (BGP)* abgelöst

Die Unterscheidung der beiden Protokollarten hat seine Ursache in den zum Teil sehr unterschiedlichen Anforderungen an das jeweilige Protokoll. So sollten Routen, die ein EGP wählt, vor allem stabil und sicher sein, ein IGP dagegen sollte immer in der Lage sein, den besten Weg zu wählen. Zur Bestimmung der optimalen Route sind mehrere Metriken im Einsatz. Eine davon ist die Anzahl der Hops (Sprünge), die bis zum Zielrechner zu überwinden sind. Dabei zählt jeder Router auf dem Weg als ein Hop. Ein Rechner im eigenen Netz hat also einen Hop-Count von 0, ein Rechner in einem angrenzenden Netz, der über den lokalen Router erreicht werden kann, hat einen Hop-Count von 1. Andere Metriken wären z.B. die Auslastung der Netzwerkverbindungen oder die Zeit bis zum erreichen des Zielrouters.

Durch ihre Arbeitsweise können **Router** auch die Aufgabe der **Flusssteuerung** übernehmen. Diese sind im wesentlichen die Festlegung des Übertragungsweges (netzübergreifend), die alternative Wegewahl sowie die Übertragung von Schicht-3-Datenpaketen. Router sind beispielsweise in der Lage, bei Überlast oder Ausfall einer Übertragungsstrecke redundante Wege zur Übertragung der Datenpakete zu verwenden.

Im Vergleich zu Bridges sind Router wesentlich teurer, langsamer und schwieriger zu installieren und konfigurieren, haben aber die oben beschriebenen Vorteile des Netzmanagements. ([3], Kap. 20.3)

Router wurden früher - und manchmal auch heute noch - Gateways genannt. Heute wird dieser Begriff jedoch bei der Verknüpfung und einer entsprechenden Übersetzung von verschiedenen Protokollen auf höherer Ebene verwendet.

### 3.5.1 Routing-Algorithmen

Während statische Routen für lokale Netze durchaus ausreichend sein können, da hier selten Änderungen der Netzwerkstruktur auftreten, stellt ein Netzwerk wie das Internet mit einem ständigen Wechsel von verfügbaren Verbindungen andere Anforderungen. Hier kommen sogenannte adaptive oder dynamische Routing-Algorithmen zum Einsatz. Die Routing-Entscheidung wird aufgrund von Parametern wie Verfügbarkeit, Geschwindigkeit und Struktur des Netzes getroffen. Dabei haben zwei Algorithmen Bedeutung erlangt:

- *Distance-Vector Routing:* Bei diesem Algorithmus tauscht jeder Router seine Routing-Tabelle mit seinen Nachbarn aus. In dieser Tabelle sind alle Netze vermerkt, mit denen der Router direkt verbunden ist. Die Entfernung zu den Netzen wird in Hops angegeben, die als Metrik für die Bestimmung der optimalen Route benutzt werden. Der Austausch der Tabellen wird in bestimmten Zeitabständen wiederholt und jeder Router aktualisiert aufgrund der empfangenen Tabellen seine eigene Tabelle. Wird dabei eine Route ermittelt, die weniger Hops zu einem Ziel enthält als eine bereits bestehende, wird diese Route aktualisiert. Wenn alle Routing Tabellen auf dem gleichen Stand sind, haben alle Router die gleiche Sicht auf das gesamte Netzwerk.

  Die Ansprüche an den Router sind bei diesem Algorithmus recht gering, da er nur den jeweils nächsten Hop kennen muss. Allerdings stabilisieren sich die Routing-Tabellen im Netz nur langsam, wenn sich Veränderungen in den Routen ergeben; die Tabellen konvergieren also langsam.

- *Link-State Routing:* Bei diesem Algorithmus werden nicht die gesamten Routing-Tabellen ausgetauscht, statt dessen verschickt jeder Router in gewissen Zeitabständen Informationen, welche anderen Router er direkt erreichen kann. Um den Zustand dieser Verbindungen zu überprüfen, sendet der Router periodisch Nachrichten an seine Nachbarn. Die dadurch erhaltenen Informationen über Zustand der Verbindung und die Nachbarn werden dann an alle Router verschickt, die aus diesen Informationen eine Sicht auf das gesamte Netzwerk aufbauen und die optimale Route nach dem *Shortest Path First* Algorithmus berechnen.

Das Link-State Routing stellt im Gegensatz zum Distance-Vector Routing sehr hohe Ansprüche an Rechenleistung und Speicherressourcen der Router. Dafür stabilisieren sich die Routing Tabellen bei einer Veränderung aber schneller, da nicht die gesamten Routing-Tabellen übertragen werden müssen und es nur einen Durchlauf der Informationen durch das gesamte Netz benötigt.

### 3.5.1.1 Das Count-to-Infinity Problem

Neben den Nachteilen des Distance-Vector Routing, die bereits beschrieben wurden, ist in diesem Algorithmus ein prinzipielles Problem vorhanden. Das Problem soll anhand eines Teilnetz mit fünf Knoten, bei dem die Metrik die Anzahl der Teilstrecken ist, anschaulich gemacht werden.

A sei anfänglich ausgeschaltet und dies ist allen Routern bekannt, daher haben alle Router die Verzögerung nach A auf unendlich gesetzt.

Schaltet sich A zum Zeitpunkt t = 0 wieder an, erfahren dies alle anderen Router über den Vektoraustausch. Der Einfachheit halber nehmen wir an, das der Vektoraustausch zwischen allen Routern immer zum gleichen Zeitpunkt erfolgt.

Zum Zeitpunkt t = 1 findet der erste Vektoraustausch statt und Router B aktualisiert seine Routingtabelle. Er vermerkt, dass A eine Teilstrecke nach links entfernt ist. Alle anderen Router denken noch, dass A nicht erreichbar ist.

Beim nächsten Austausch erfährt C, dass B einen Pfad der Länge 1 zu A hat und aktualisiert seine Routingtabelle auf eine Pfadlänge von 2. Beim nächsten Austausch erfährt dies D, und dann schließlich E.

Die Nachricht verbreitet sich also mit einer Geschwindigkeit von einer Teilstrecke pro Austausch. In einem Teilnetz mit N Teilstrecken tauschen alle Router innerhalb von N die Nachrichten über neu hinzugekommene Router und Leitungen aus.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| t = 0 | | ∝ | ∝ | ∝ | ∝ |
| t = 1 | | 1 | ∝ | ∝ | ∝ |
| t = 2 | | 1 | 2 | ∝ | ∝ |
| t = 3 | | 1 | 2 | 3 | ∝ |
| t = 4 | | 1 | 2 | 3 | 4 |

*Bild 3-16 Verhalten bei der Aktivierung eines Knotens*

Nun seien alle Router betriebsbereit, die Router B, C, D und E haben die Entfernung 1, 2, 3 bzw. 4 zu A. Wir nehmen an, das A ausfällt oder die Verbindung zwischen A und B unterbrochen wird.

Beim ersten Austausch erfährt B nichts von A. C teilt aber mit, dass er einen Pfad der Länge 2 zu A hat. B weiß nicht, dass dieser Pfad über ihn, B, zu A führt. Deshalb denkt B, er könne A über C mit einer Pfadlänge von 3 erreichen und aktualisiert seine Tabelle.

Beim nächsten Austausch erfährt C, dass alle seine Nachbarn behaupten, einen Pfad der Länge 3 zu A zu haben. Er wählt einen aus und setzt seine Entfernung zu A auf 4. Dieses Spiel setzt sich, wie in der Abbildung gezeigt, weiter fort, bis irgendwann alle Router den Wert ∝ für die Entfernung zu A in der Tabelle haben. Wann dies geschieht, hängt vom numerischen Wert von ∝ ab.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| t = 0 | | 1 | 2 | 3 | 4 |
| t = 1 | | 3 | 2 | 3 | 4 |
| t = 2 | | 3 | 4 | 3 | 4 |
| t = 3 | | 5 | 4 | 5 | 4 |
| t = 4 | | 5 | 6 | 5 | 6 |
| t = 5 | | 7 | 6 | 7 | 6 |
| t = 7 | | 7 | 8 | 7 | 8 |
| ... | | ... | ... | ... | ... |
| t = ? | | ∝ | ∝ | ∝ | ∝ |

*Bild 3-17 Verhalten bei Ausfall einer Strecke*

Aus der Abbildung wird deutlich, dass sich schlechte Nachrichten langsam verbreiten. Die Zahl der benötigten Übertragungen hängt davon ab, auf welchen Wert ∝ gesetzt wurde. Daher wird ∝ auf den längsten Pfad + 1 gesetzt; wenn allerdings die Verzögerung als Metrik benutzt wird, gibt es keine definierte Obergrenze. Hier ist ein hoher Wert notwendig, damit von einem Pfad mit hoher Verzögerung nicht angenommen wird, er sei ausgefallen.

### 3.5.2 Routing-Protokolle

Das bekannteste Routing-Protokoll stellt wohl das im RFC 1058 definierte Router Information Protocol (RIP) dar; dieses Protokoll ist als Standard-Routingprotokoll auf allen UNIX-Systemen verfügbar (implementiert durch den Routing-Daemon *routed*). RIP arbeitet mit den Distance-Vector Routing und benutzt als Metrik die Anzahl der Hops, wobei die maximale Anzahl der Hop-Counts bei 15 liegt. Ein Hop-Count von 16 bedeutet, dass das Netz nicht erreichbar ist. Mit einer RIP-Nachricht können bis zu 25 Routen weitergegeben werden, so dass für die komplette Übertragung einer Routing-Tabelle oft mehrere Nachrichten benötigt werden. Als Erweiterung zum Distance-Vector Routing kommen Verfahren wie *Split Horizont* (Informationen über das Netzwerk des benachbarten Routers müssen nicht an diesen weitergegeben werden) oder *Triggered Updates* (gezielte Änderung der Tabellen, wenn sich eine Route ändert) zum Einsatz, damit sich die Informationen in den Routern schnell stabilisieren und keine falschen Routen weitergegeben werden.

Das OSPF-Protokoll (OSPF= Open Shortest Path First), das im RFC 1247 beschrieben ist, nutzt hingegen Link-State Routing. Darüber hinaus berücksichtigt OSPF noch weitere Parameter bei der Bestimmung der optimalen Route (z.B. kann eine Route nach den Kosten beurteilt werden, wobei die Kosten sich etwa an Bandbreite, Verzögerung und Zuverlässigkeit orientieren).

Nachfolgende Tabelle stellt die Eigenschaften der (transparenten) Bridge denen des Routers gegenüber. ([3], Kap. 20.3-20.4)

| **Bridges** | **Router** |
|---|---|
| • hören in das LAN | • hören in das gesamte Netzwerk |
| • kennen das LAN | • kennen das gesamte Netzwerk |
| • erstellen eine Netzwerktabelle | • erstellen eine Routing-Tabelle |
| • filtern lokalen Verkehr | • filtern lokalen Verkehr nicht, da sie vom Anwender direkt aufgefordert werden, wenn sie Daten weiterleiten sollen |
| • reichen nichtlokalen Verkehr weiter | • geben nichtlokalen Verkehr gezielt weiter |
| • alternative Wegewahl nicht möglich | • alternative Wegewahl möglich |

## 3.6 Gateway

Durch ein Gateway werden Netzwerke ab der Transportschicht (Schicht 4) oder höher miteinander gekoppelt. In der Regel werden Netzwerke miteinander verbunden, die überhaupt nichts mehr gemeinsam haben.

Aus diesem Grund erstreckt sich die Aufgabe eines Gateways in den meisten Fällen über alle sieben Ebenen des OSI-Referenzmodells. Gateways verbinden also Netze mit unterschiedlichen Protokollfamilien wie z.B. ein TCP/IP-Netz mit einem DECnet. Sie müssen softwaremäßig den Übergang zwischen den verschiedenen Protokollen bewerkstelligen. Zusätzlich zur Protokollumwandlung führen sie noch Codekonvertierungen durch, außerdem erfüllen sie die Aufgaben eines Routers. Beim Einsatz eines Gateways geht die End-zu-End-Kontrolle der Transportverbindung verloren, was von der ISO eigentlich nicht vorgesehen ist, sich aber auch nicht vermeiden lässt, wenn sich die zu verbindenden Netze auf der Transportschicht oder höher unterscheiden.

Da die **Umwandlung von Protokollen meistens sehr aufwendig** ist, beschränken sich die meisten Gateways bezüglich ihrer Anwendungen auf **Terminalemulations- und Filetransfer-Fähigkeiten**. Aufgrund der komplexeren Aufgaben arbeitet ein Gateway im Vergleich zu Bridges und Routern mit geringerer Geschwindigkeit.

([1], S.17-18; [3], Kap. 20.5; [7], S.17)

**Application Gateway**

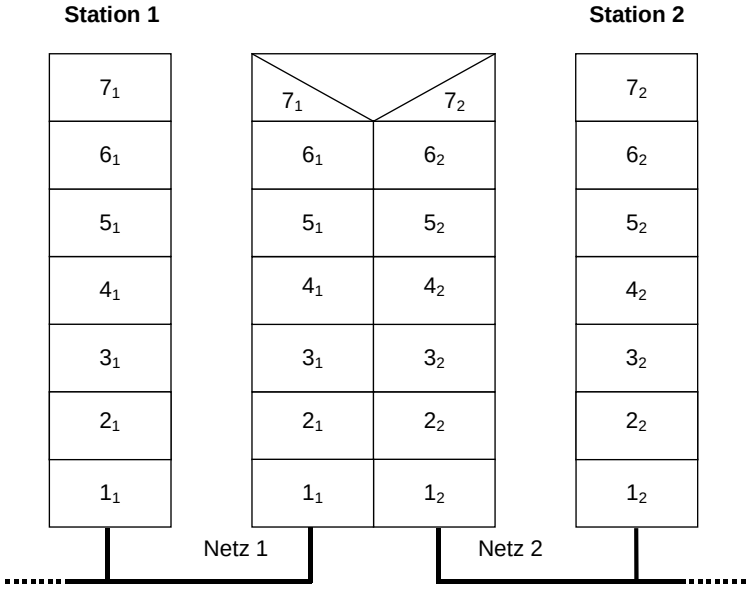



*Bild 3-18 Netzkopplung über ein Application Gateway*

Mittels eines Application Gateways werden einzelne Anwendungen aufeinander abgebildet. Die erreichbare Funktionalität des Gateways kann maximal aus dem Schnitt der Funktionalität der beiden zu verbindenden Anwendungen bestehen. Daher ist ein solches Gateway im Regelfall nie transparent für einen Benutzer, denn er verliert gewohnte Teile seiner Funktionalität und er muss ggf. auch noch gewisse Konventionen und Namensfestlegungen der anderen Anwendung kennen. Der

Unterhalt von Application Gateways stellt sowohl von der Performance her als auch vom Know-how des Betreibers gewisse Anforderungen. Daher werden solche Gateways meist an zentraler Stelle unterhalten. Eine Anpassung bei den Endpunkten der Verbindung ist zumeist nicht erforderlich - höchstens zur Verbesserung des Bedienungskomforts des Endbenutzers. ([2], S.195)

## 3.7 Gegenüberstellung

In nachfolgender Tabelle werden die oben beschriebenen Netzkoppeleinheiten anhand ihrer wichtigsten Merkmale gegenübergestellt, um die Unterschiede noch einmal zu verdeutlichen.

| | **Repeater** | **Bridge/Switch** | **Router** | **Gateway** |
|---|---|---|---|---|
| koppelt | Netze gleichen Typs<br><br>(Bsp.: Ethernet mit Ethernet) | Netze gleichen Typs<br><br>(Bsp.: Ethernet mit Ethernet) | Netze mit unterschiedlichen Netzadressen<br><br>(Bsp.: TCP/IP-LAN1 mit TCP/IP-LAN2) | unterschiedlichste Netzwerke<br><br>(Bsp.: TCP/IP mit DECnet) |
| Kopplungsschicht | Bitübertragungsschicht<br><br>(Schicht 1) | Sicherungsschicht<br><br>(Schicht 2) | Vermittlungsschicht<br><br>(Schicht 3) | ab Transportschicht aufwärts<br><br>(Schichten 4-7) |
| Anzahl der zu koppelnden Netze begrenzt ? | ja | nein | nein | nein |
| Für die Anwendung transparent ? | ja | ja | nein | nein |
| Unterstützt alternative Wegewahl ? | nein | nein | ja | ja |
| Unterstützt Filtering ? | nein | ja | nein (da nicht transparent) | nein (da nicht transparent) |
| protokollabhängig ? | nein | nein | in der Regel ja | ja |

Nachfolgende Graphik zeigt ein Beispiel für den Einsatz verschiedener Netzkoppeleinheiten. Damit soll verdeutlicht werden, welche Netze mit welchen Netzkoppeleinheiten verbunden werden können.

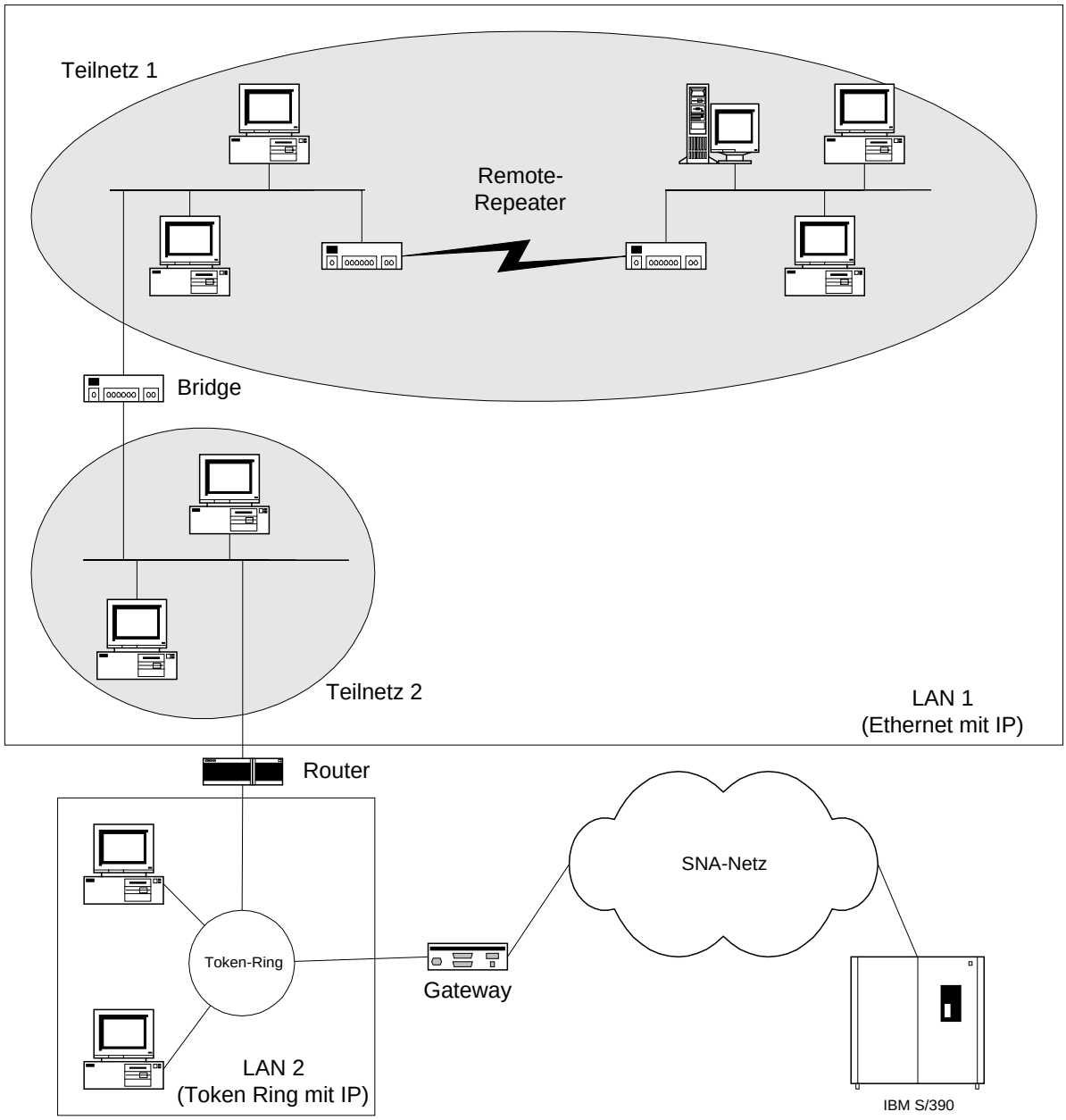



*Bild 3-19 Beispiel für den Einsatz verschiedener Netzkoppeleinheiten*

## 3.8 Aufgaben

1 Nennen Sie drei grundsätzliche Aufgaben eines Netzkoppelelementes und erläutern Sie kurz diese Aufgaben.

2 a) Erklären Sie anhand einer Skizze unter Verwendung der ISO/OSI-Schichten, was ein Repeater ist.

b) Erklären Sie anhand einer Skizze unter Verwendung der ISO/OSI-Schichten, was eine Bridge ist.

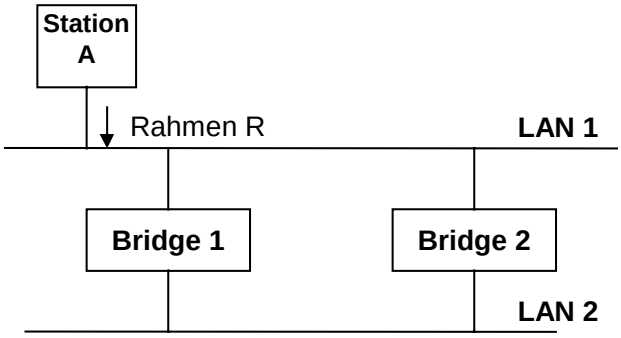
c) Erklären Sie anhand einer Skizze unter Verwendung der ISO/OSI-Schichten, was ein Router ist.

3 Wann wird zur Kopplung zweier Netze ein Gateway eingesetzt ? Nennen Sie ein Beispiel.

4 Erläutern Sie die Unterschiede und die Gemeinsamkeiten einer filternden Bridge und eines Routers.

5 a) Welches Problem ergibt sich bei Verwendung redundanter (paralleler) transparenter Bridges ?

b) Erläutern Sie anhand des in der Abbildung gezeigten Beispiels, wie dieses Problem entsteht.




c) Wie heißt der von IEEE genormte Algorithmus zum Verhindern dieses Problems ?

d) Wie wird das Problem vermieden ?

6 a) Welche beiden grundsätzlichen Strategien werden beim Switching angewandt ? Erläutern Sie diese Strategien kurz.

b) Was ist der Vorteil und der Nachteil der beiden Strategien ?

c) Durch welches Verfahren werden die Vorteile der beiden Strategien erhalten und die Nachteile weitestgehend vermieden ? Nennen Sie den Namen des Verfahrens und erläutern Sie dieses kurz.

7 a) Was ist die Gemeinsamkeit zwischen einem Repeater und einer nichtfilternden Bridge, was ist ihr Unterschied ?

b) Was ist die Gemeinsamkeit zwischen einer nichtfilternden Bridge und einer filternden Bridge. Was ist ihr Unterschied ?

c) Mit welchen Schicht-Adressen (Schichten im Sinne von ISO) arbeitet eine Bridge, mit welchen Schicht-Adressen ein Router ?

8 a) Sie sollen zwei Lokale Netze koppeln. Das eine ist ein Token Ring, das andere ein Ethernet. Als Koppelelemente stehen zur Auswahl: Repeater, Bridge, Router. Begründen Sie, welches Koppelelement Sie warum einsetzen und welche warum nicht!

b) Zeichnen Sie ein Schichtenmodell für das Koppelelement, eine Station am Ethernet und eine Station am Token Ring. Erklären Sie Ihre Skizze!

9 Zwei Rechner am Netzrand kommunizieren über ein Fernmeldenetz. Wie viele ISO-Schichten hat ein Vermittlungsrechner (Relais, Transitsystem) ? Warum hat er diese Anzahl von Schichten ?

10 Wozu dient ein Spanning Tree Algorithmus ?

11 Welches Problem tritt auf, wenn ein Token Ring und ein Ethernet mittels einer Mixed-Media-Bridge miteinander verbunden werden ?

12 Source-Route-Bridging

a) Erläutern Sie, wodurch beim Source-Route-Bridging der Weg des Paketes zum Empfänger festgelegt ist.

b) Wie wird der Weg vom Sender zum Empfänger ermittelt ?

c) Nennen Sie einen Vorteil und einen Nachteil dieses Verfahrens.

13 Routing

a) Welche Arten von Routing-Protokollen werden grundsätzlich unterschieden und wo werden sie eingesetzt ?

b) Nennen Sie die beiden in der Vorlesung behandelten Routing-Algorithmen und erläutern Sie, wie bei dem jeweiligen Algorithmus der Austausch der Routing-Informationen zwischen den Routern erfolgt und in welchem bekannten Routing Protokoll der jeweilige Algorithmus eingesetzt wird.

# 4 Die TCP/IP-Protokollfamilie

Die Bezeichnung TCP/IP steht für eine Protokollfamilie, deren Ursprünge in der Forschungsarbeit der Advanced Research Projects Agency (ARPA), einer Forschungseinrichtung des US Militärs, liegen. Im gleichen Maß, wie das INTERNET an Bedeutung gewann, nahm auch die Bedeutung von TCP/IP zu. Die Entwicklung begann in den Jahren 1977-79, als die Architektur und die Protokolle definiert wurden. Den ersten Einsatz fanden die Protokolle im bekannten ARPANET, das im Jahre 1980 auf TCP/IP umgestellt wurde und sich schnell zum ersten Backbone des Internet entwickelte. Das ARPANET verband amerikanische Forschungs- und Militäreinrichtungen, bevor es in einen militärischen Teil, das MILNET, und einen Teil für die Forschungseinrichtungen, der den Namen ARPANET behielt, aufgeteilt wurde.

Einen großen Anteil am Erfolg von TCP/IP hatte die Integration dieser Protokolle in die UNIX-Version der University of California, der Berkeley Software Distribution (BSD UNIX). Neben einigen nützlichen Utilities wurde in dieser UNIX-Version auch erstmals die Programmierschnittstelle Socket eingeführt. Im Laufe der Zeit fanden die TCP/IP-Protokolle auch Verbreitung außerhalb der Forschungseinrichtungen, nicht zuletzt durch die rasante Verbreitung des INTERNET, das aus dem ARPANET hervorging. Dadurch ist heute für jedes Betriebssystem mindestens eine Implementierung des TCP/IP-Protokollstapels verfügbar und TCP/IP stellt den Standard bei der Realisierung heterogener Netze dar.

## 4.1 Standardisierung im Internet

### 4.1.1 Das Internet Architecture Board

Da TCP/IP nicht von einer Organisation oder einem Unternehmen entwickelt wird, sondern eine Vielzahl von Entwicklern ihren Beitrag dazu leisten, stellt sich die Frage, wer diese Entwicklungen koordiniert und letztendlich entscheidet, welche Entwicklungen in die offizielle TCP/IP-Protokollsuite aufgenommen werden. In der Anfangszeit nahm diese Aufgabe eine Einrichtung der ARPA wahr, das Internet Control and Configuration Board (ICCB). Das ICCB wurde 1983 neu organisiert und trägt nun den Namen **Internet Architecture Board** (IAB). Bild 4-1 zeigt den prinzipiellen Aufbau, den das IAB heutzutage hat [22].

Das IAB teilt sich in zwei große Gruppen auf:

- die **Internet Engineering Task Force**, die sich mit der kurz- und mittelfristigen Entwicklung befasst. Die IETF ist wiederum in 12 areas aufgeteilt, der chairman des IETF und die area manager bilden die Internet Engineering Steering Group (IESG), die für die Koordinierung der Anstrengungen innerhalb der IETF verantwortlich ist.
- die **Internet Research Tasking Force**, die Forschungsaktivitäten zur grundsätzlichen Struktur der Protokolle und zur Architektur koordiniert. Wie im IETF gibt es auch hier eine kleine Gruppe, die Internet Research Steering Group (IRSG), die die Aktivitäten koordiniert. Da die IRTF sehr viel kleiner als die IETF ist, gibt es hier keine Unterteilung in areas.

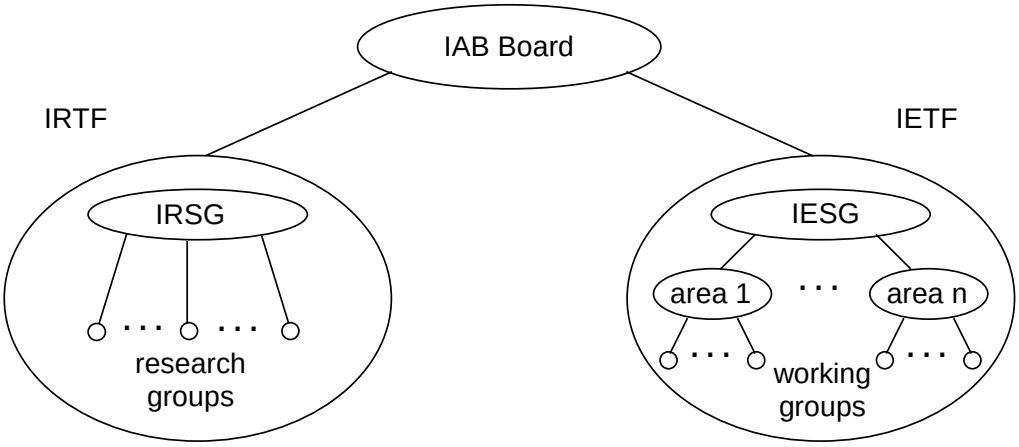



*Bild 4-1 Struktur des IAB*

### 4.1.2 Internet Request for Comments

Die Standards für alle TCP/IP-Protokolle, Vorschläge für neue oder überarbeitete Protokolle und eine Vielzahl weiterer Dinge (z.B. ein Begriffsverzeichnis in RFC1983 „Internet User's Glossary") findet man in einer Reihe von technischen Berichten, den sogenannten **Request for Comments** oder RFCs. Jedes Mitglied der Internet-Gemeinschaft kann ein Protokoll, das in der Internet-Protokollfamilie benutzt werden soll, entwickeln, dokumentieren, implementieren und testen. Die IAB verlangt lediglich, dass Protokolle in der Reihe Request For Comments dokumentiert werden. Jedes RFC Dokument bekommt eine Nummer durch den RFC-Herausgeber (einem Mitglied des IAB) zugeteilt. Falls sich der Inhalt des RFCs ändert, wird eine neue Nummer vergeben. Um Verwirrungen zu vermeiden, wird auf dem Deckblatt des neueren RFCs klar vermerkt, dass dieser RFC einen älteren ablöst.

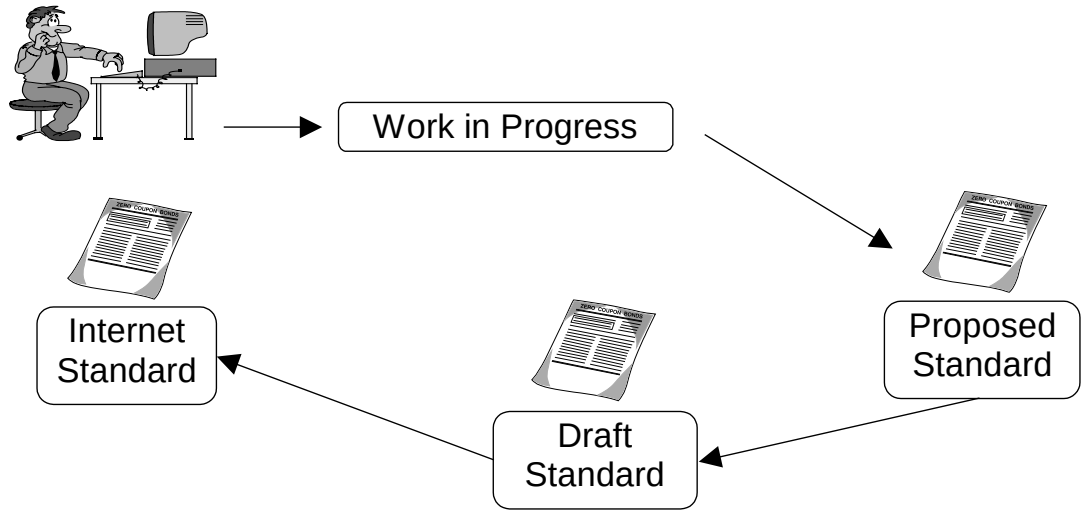



*Bild 4-2 Internet Standardisierungsprozess*

Der Weg eines RFCs von der Definition bis hin zu einem Internet-Standard geht über die folgenden vier Stationen (Bild 4-2):

Während der Entwicklungszeit besitzt das RFC den Stand "*Work in Progress*". Ist die Definition abgeschlossen, so geht das RFC in den "*Proposed Standard*" über. In dieser Zeit werden nur noch Detailfehler beseitigt, die sich erst durch die Implementierung der Spezifikation zeigen.

Der nächste Schritt auf dem Weg zum Internet-Standard ist der "*Draft Standard*", der frühestens nach 6 Monaten erreicht werden kann. In diesem Status muss das RFC wiederum 4 Monate verbleiben, bis es dann schließlich zum "*Internet-Standard*" erhoben werden kann.

RFCs, die durch ein aktuelleres RFC abgelöst werden, bekommen den Status "*Obsolete*".

## 4.2 TCP/IP-Protokollarchitektur

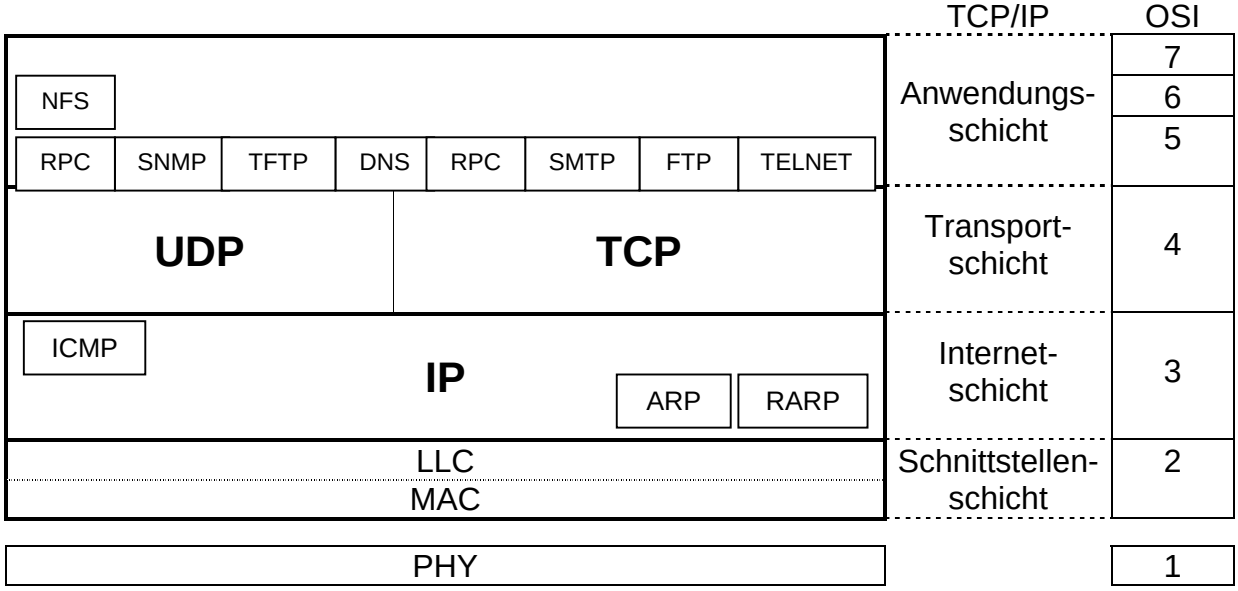



*Bild 4-3 Die TCP/IP-Protokollarchitektur*

Wie in Bild 4-3 zu sehen ist, besteht das Schichtenmodell der TCP/IP-Architektur im Gegensatz zum OSI-Schichtenmodell lediglich aus vier Schichten:

- Die **Schnittstellenschicht** (Network Level) ist für die Übertragung der Daten über ein physikalisches Netzwerk (z.B. Ethernet, Token Ring, ...) verantwortlich.
- Auf der **Internetschicht** (Internet Level) existiert nur ein einziges Protokoll, das Internet Protocol (IP). Das Internet Control Message Protocol (ICMP) ist Bestandteil jeder IP-Implementierung und transportiert Fehler- sowie Diagnose-Informationen für IP.
- Auf der **Transportschicht** (Transmission Level) werden neben dem Transmission Control Protocol (TCP) und dem User Datagramm Protocol (UDP) im Internet noch weitere Protokolle betrieben, vor allem zu Forschungszwecken. Die beiden am weitesten verbreiteten Protokolle sind jedoch TCP und UDP.
- Die **Anwendungsschicht** (Application Level) umfasst die Schichten 5-7 des OSI-Modells und enthält alle anwendungsbezogenen Protokolle für die Kommunikation zwischen Anwendungsprozessen, wie z.B. "Electronic Mail" oder "File Transfer".

## 4.3 Schnittstellenschicht

Die Schnittstellenschicht übernimmt die Übertragung der Daten über ein bestimmtes Übertragungsmedium und kapselt die Internetschicht vom Medium ab. Das IP-Protokoll ist dadurch unabhängig vom verwendeten Übertragungsmedium und muss nicht an dieses angepasst werden. Eine der großen Vorteile von TCP/IP liegt in der Flexibilität, beinahe jedes Medium für die Übertragung benutzen zu können. So können IP-Pakete beispielsweise über Ethernet, Token Ring, ATM, X.25 und viele weitere Netze transportiert werden.

In den weiteren Kapiteln wird aufgrund der großen Verbreitung hauptsächlich auf die Verwendung von TCP/IP über Ethernet (IEEE 802.3) eingegangen.

| 6 | 6 | 2 | 46 - 1500 | 4 |
|---|---|---|---|---|
| DA | SA | TYPE/Length | MAC-Data | FCS |

*Bild 4-4 Ethernet-Rahmenformat*

Bild 4-4 zeigt den Aufbau eines Ethernet MAC-Frames. Der Frame enthält folgende Felder:

- die Ethernet Adresse des Rechners, an den der Frame geschickt werden soll (Destination Address, MAC- oder auch Hardware-Ziel-Adresse, bestehend aus 48 Bit = 6 Octets[2]), bei einem MAC-Broadcast enthält dieses Feld den Wert FFFFFFFF
- die Ethernet Adresse des Rechners, von dem der Frame abgeschickt wurde (Source Address, MAC- oder auch Hardware-Quell-Adresse),
- ein Typ-Längenfeld, dass entweder die Länge des Datenfeldes oder einen Identifier für die transportierten Daten enthält, z.B.:

  0800h: IP-Datagramm

  0806h: ARP-Paket

  0835h: RARP-Paket

- die Nutzdaten im MAC-Data-Feld. In diesem Feld sind im Normalfall wiederum PDUs höherer Schichten enthalten, bei TCP/IP also ein IP-Paket (oder ein Teil eines IP-Paketes)
- eine Prüfsumme (Frame Check Sequence FCS), die das gesamte Paket absichert.

## 4.4 Internet-Schicht

### 4.4.1 IP – Internet Protocol

Derzeit aktuelle Version: IPv4, definiert in RFC 760 (Sept. 1981)

Das Internet-Protokoll bietet den Protokollen der Transportschicht einen verbindungslosen, unzuverlässigen Paketübermittlungsdienst. Hauptaufgabe von IP sind die Adressierung von Rechnern, das Fragmentieren von Paketen und das Routing der Pakete. Es enthält keine Funktion für die End-zu-End-Sicherung von Nachrichten oder für die Flusskontrolle. Pakete werden so gut wie möglich übertragen (Best Effort) - garantiert ist die Zustellung allerdings nicht.

[2] Im Netzbereich wird meist der Begriff Octet an Stelle von Byte verwendet, da Byte eine hardwareabhängige Größe von Zeichen darstellt, Octet jedoch immer einen 8 Bit großen Speicherplatz bezeichnet

Jedes IP-Datagramm wird als einzelnes Paket, völlig unabhängig von anderen Datagrammen, durch das Netz zum Empfänger übertragen. Für jedes Datagramm wird innerhalb des Netzes der optimale Weg ermittelt. Dabei können sich Datagramme auf dem Weg zum Empfänger überholen und dadurch in geänderter Reihenfolge beim Empfänger eintreffen. Die Aufgabe, die Pakete in die richtige Reihenfolge zu bringen, muss die Transportschicht übernehmen.

IP stellt keine gesicherte Verbindung zur Verfügung, sondern verlässt sich darauf, dass die Protokolle der höheren Ebenen die End-zu-End-Kontrolle gewährleisten. IP ist nicht in der Lage, verlorene oder von der Schnittstellenschicht abgelehnte Datagramme neu zu generieren und erneut zu übertragen.

Das IP-Protokoll ist, wie die meisten Standard-Protokolle, sehr genau definiert. Die Definitionen beschreiben das Format des IP-Datagramms, die einzelnen Felder im Header und den Ablauf der Datenübermittlung.

#### 4.4.1.1 IP-Adressen

Zur Adressierung eines Kommunikationspartners kommen auf IP-Ebene 32-Bit lange Adressen (IP-Adressen) zum Einsatz. In diesem Zusammenhang ist wichtig, dass eine IP-Adresse die Verbindung eines Rechners zum Netz identifiziert, nicht den Rechner selbst. Dies hat zur Folge, dass ein Rechner, der gleichzeitig an zwei oder mehr Netzen angeschlossen ist (ein sogenannter *multihomed host*), auch mehrere IP-Adressen benötigt. Eine weitere Folge ist, dass ein Rechner, der in ein anderes Netz verlegt wird, auch eine neue IP-Adresse benötigt.

Um das Routing der Pakete möglichst effizient gestalten zu können, wurden die verfügbaren Adressen in mehrere Klassen aufgeteilt:

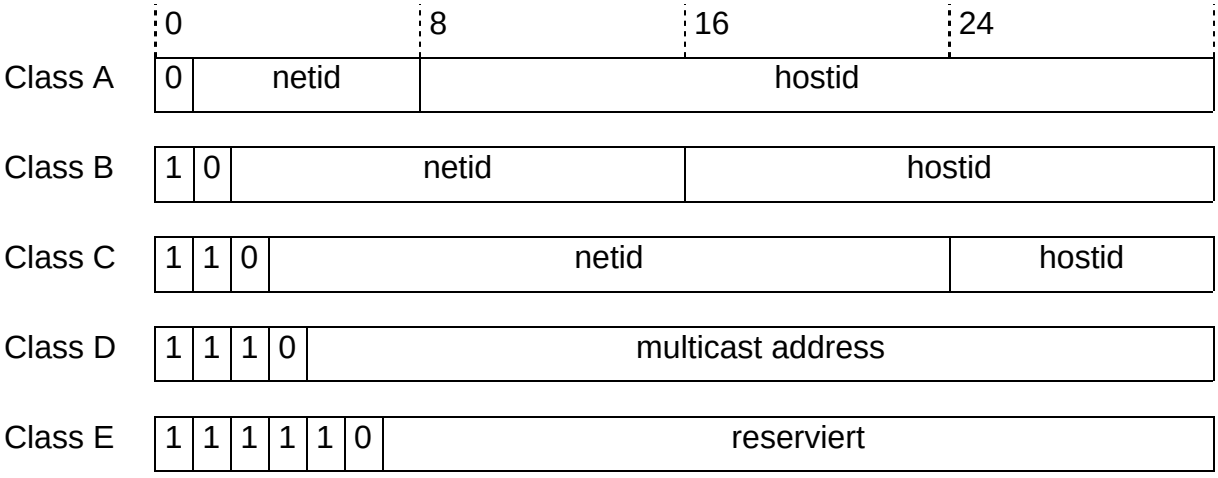



*Bild 4-5 IP-Adressklassen*

Jede Adresse besteht aus zwei Teilen: einem Teil, der das Netzwerk bezeichnet, in dem sich der Rechner befindet (netid), und einem weiteren Teil, der den Rechner adressiert (hostid).

Die Adressklassen D und E sind für besondere Anwendungen reserviert (z.B. Class D-Adressen für Multicast, beispielsweise im Mbone) und werden deswegen nicht näher betrachtet.

In den Klassen A, B und C ist die netid entweder 8, 16 oder 24 Bit lang, woraus sich auch die Zahl der Netze und der Rechner in einem Netz ergibt:

| | |
|---|---|
| **Class A:** | Eine Class A-Adresse besteht aus 8 Bit Netzadresse und 24 Bit Netzknotenadresse. Das höchstwertige Bit der Netzadresse hat immer den Wert "0". Eine Class A-Adresse kann deshalb im ersten Octet Werte zwischen 0 und 127 annehmen.<br><br>Damit können theoretisch 128 unterschiedliche Class A-Netze mit jeweils $2^{24}$ = 16777216 Netzknoten gebildet werden. Die Netzadresse 127 bildet jedoch eine Ausnahme. Dieses Class A-Netz ist für die *"Loopback"* Funktion aller Netzknoten (unabhängig von der jeweiligen Netzklasse) reserviert. Wird ein Datagramm an einen Knoten im Netz 127 adressiert, wird das Datenpaket innerhalb des Sendeknotens an den Sender zurückgeschickt. Zu keinem Zeitpunkt darf ein Datagramm mit der Netzadresse 127 auf das Netz übertragen werden, diese Adressen werden im lokalen Rechner zu Testzwecken oder zur Interprozesskommunikation eingesetzt. |
| **Class B:** | Class B-Adressen bestehen aus 16 Bit Netzadresse und 16 Bit Netzknotenadresse. Die zwei ersten Bits höchster Ordnung in der Netzadresse haben den Wert "10". Class B-Adressen ermöglichen die Definition von 16384 Netzen mit jeweils 65536 Netzknoten. Der Wertebereich des ersten Octets liegt zwischen 128 und 191. |
| **Class C:** | Der dritte Adresstyp, Class C, besteht aus 24 Bit Netzadresse und 8 Bit Knotenadresse. Die drei Bits höchster Ordnung in der Netzadresse haben den Wert "110". Eine Class C-Adresse erlaubt 2097152 Netze mit jeweils 256 Netzknoten. Der Wertebereich des ersten Byte beträgt 192 bis 223. |

*Bild 4-6 Übersicht über die Adressklassen*

Nicht alle möglichen IP-Adressen dürfen auch vergeben werden; vielmehr gilt es einige Besonderheiten und Einschränkungen zu beachten, die im folgenden erläutert werden:

- Es dürfen nicht alle Bits eines Feldes (netid oder hostid) “0“ sein, da dies im Falle der netid immer das eigene Netz adressiert und im Falle der hostid die Adresse des gesamten Netzes darstellt. Die IP-Adresse 134.108.0.0 beispielsweise darf keinem Host zugeordnet werden, da diese Adresse das gesamte Netz 134.108 bezeichnet. Ein Sonderfall ist der Fall, in dem alle Bit der netid und der hostid “0“ sind (0.0.0.0), dies kann folgende Bedeutungen haben:
  - Speziell bei Routern bezeichnet dieser Eintrag in der Routing-Tabelle die Standard-Route, also die Route, an die alle Pakete geschickt werden, für die keine andere Route eingetragen ist
  - diese Adresse wird von plattenlosen Systemen verwendet, die beim Booten noch keine IP-Adresse haben und diese zunächst vom Server beziehen müssen
  - außerdem ist diese Adresse eine veraltete Broadcastadresse, die einer fehlerhaften Implementierung in Berkeley UNIX entstammt

- Es dürfen nicht alle Bits eines Feldes (netid oder hostid) “1“ sein, da dies für Broadcasts reserviert ist. Die Adresse 134.108.255.255 ist also die Broadcast-adresse des Netzes 134.108 und adressiert alle Rechner in diesem Netz. Auch hier stellt der Fall, dass alle Bits der netid und der hostid 1 sind (255.255.255.255) einen Sonderfall dar; dies adressiert alle Hosts eines Netzes, stellt also eine limited oder local Broadcastadresse dar.

Zusammenfassend gilt also:

Eine IP-Adresse, bei der alle Bits der hostid “0“ sind, bezeichnet nicht einen einzelnen Rechner, sondern ist reserviert, um das gesamte IP-Netz zu bezeichnen.

Sind alle Bits der hostid “1“, ist diese Adresse die Broadcastadresse des Netzes und adressiert alle Rechner in diesem Netz.

Notiert werden IP-Adressen meist als 4, durch einen Punkt getrennte Dezimalzahlen, diese Darstellung wird auch als dotted decimal notation bezeichnet. Jede Dezimalzahl repräsentiert dabei 8 Bit der IP-Adresse, kann also einen Wert zwischen 0 und 255 annehmen.

Beispiel: 10000110 01101100 00111000 00111100 = 134.108.56.60

Manchmal werden IP-Adressen auch in einer Hexadezimalen Schreibweise angegeben, die obige Adresse würde dann als 0x86.6C.38.3C geschrieben. Diese Notation hat den Vorteil, dass mit etwas Übung leicht erkannt werden kann, welche Bits 1 und welche 0 sind.

Die folgende Zusammenfassung zeigt die verfügbaren Adressbereiche für die einzelnen Adressklassen in der dotted decimal notation [22]:

| Klasse | niedrigste Adresse | höchste Adresse |
|---|---|---|
| A | 1.0.0.0 | 126.255.255.255 |
| B | 128.0.0.0 | 191.255.255.255 |
| C | 192.0.0.0 | 223.255.255.255 |
| D | 224.0.0.0 | 239.255.255.255 |
| E | 240.0.0.0 | 247.255.255.255 |

Um sicherzustellen, dass die netid eines an das INTERNET angeschlossenen Netzes auf der ganzen Welt eindeutig ist, werden IP-Adressen von einer zentralen Organisation, der Internet Assigned Number Authority (IANA) vergeben. Wenn eine Organisation ihr Netz an das INTERNET anschließen möchte, muss sie für dieses Netz IP-Adressen vom Internet Network Information Center (INTERNIC) erwerben. Die Adressen innerhalb des erworbenen Adressbereiches kann jede Organisation selbst vergeben, allein der Teil der Adresse, der das Netz bezeichnet, wird von der IANA vergeben.

Aufgrund der begrenzten Anzahl der zur Verfügung stehenden Class A- und Class B-Adressen sind derzeit fast nur noch Class C-Adressen erhältlich, so dass auch Organisationen mit mehr als 254 Rechnern für ihr Netz Class C-Adressen zugeteilt bekommen.

Neben den offiziellen Adressen sind einige Adressbereiche für den privaten Gebrauch reserviert und werden nicht offiziell vergeben. Diese Adressen sollten immer verwendet werden, wenn für ein nicht am Internet partizipierendes Netz IP-Adressen benötigt werden.

In RFC 1918 - Address Allocation For Private Intranets – sind die folgenden Adressbereiche für den privaten Gebrauch spezifiziert worden:

| | | |
|---|---|---|
| 10.0.0.0 | - | 10.255.255.255 (Ein Klasse A-Netz) |
| 172.16.0.0 | - | 172.31.255.255 (16 Klasse B-Netze) |
| 192.168.0.0 | - | 192.168.255.255 (255 Klasse C-Netze) |

Soll das Netz zu einem späteren Zeitpunkt an das Internet angeschlossen werden, so kann dies ohne eine Änderung der bereits vergebenen IP-Adressen geschehen. Möglich wird dies durch eine Funktion namens „Network Adress Translation (NAT)“. Das Gateway, das die Verbindung zum Internet darstellt, übersetzt dabei die internen, privaten Adressen in eine offizielle IP-Adresse.

#### 4.4.1.2 Subnetze und Netzmasken

Wie aus der Übersicht in Bild 4-6 zu entnehmen ist, können in einem Class A Netz $2^{24}$ = 16777216 Rechner angeschlossen werden. Es ist leicht einzusehen, dass dies nicht sinnvoll ist, da alle diese Rechner an einem physikalischen Netzwerk angeschlossen sein müssten (ein Router setzt ja bereits zwei IP-Netze voraus). Daher gibt es die Möglichkeit, ein bestehendes Netz weiter in sogenannte Subnetze zu unterteilen. Diese Möglichkeit wird als subnetting bezeichnet; zur Unterteilung wird ein Teil der hostid als Bezeichner für ein Subnetz (subnetid) benutzt. Wie viele Bits der hostid als subnetid benutzt werden sollen, kann der Betreiber des Netzes selbst festlegen. Es ist allerdings nicht mehr möglich, anhand der Adressklasse zu unterscheiden, welcher Teil der Adresse das Netz bezeichnet.

Um festzulegen, welcher Teil der IP-Adresse zur netid gehört, wird die Netzmaske oder auch Subnetzmaske benötigt. Möchte man beispielsweise bei einem Class A-Netz das komplette zweite Octet dazu benutzen, Subnetze zu bilden, so muß die Netzmaske den Wert 255.255.0.0 haben. Durch eine logische UND Verknüpfung der Netzmaske mit einer IP-Adresse kann auf einfachem Weg die IP-Adresse in hostid und netid zerlegt werden. Bild 4-7 zeigt dies am Beispiel eines Klasse A Netzes (9.0.0.0, IBM), das mit Hilfe der Netzmaske 255.255.0.0 in 256 Subnetze unterteilt wird.

| | | | | | |
|---|---|---|---|---|---|
| IP-Adresse | 00001001 | 00100110 | 10011101 | 00010101 | = 9.38.157.21 |
| Netzmaske | 11111111 | 11111111 | 00000000 | 00000000 | = 255.255.0.0 |
| netid = | | | | | |
| IP-Adresse UND Netzmaske | 00001001 | 00100110 | 00000000 | 00000000 | = 9.38.0.0 |
| hostid | 00000000 | 00000000 | 10011101 | 01110110 | = 0.0.157.21 |

*Bild 4-7 Ermittlung der netid aus IP-Adresse und Netzmaske*

Durch die UND-Verknüpfung der IP-Adresse mit der Netzmaske wird die netid ermittelt, die hostid kann durch eine UND-Verknüpfung der IP-Adresse mit der invertierten Netzmaske ermittelt werden.

**Beispiel:**

Ein Class B-Netz mit der IP-Adresse 134.108.0.0 soll mit Hilfe des subnetting in 8 Subnetze unterteilt werden.

Lösung: Um die Class B-Adresse in 8 Subnetze zu unterteilen, müssen 3 Bit der hostid als subnetid benutzt werden. Die Netzmaske muss also die 16 Bit der Class B Adresse und zusätzlich 3 Bit der hostid maskieren. Das Netz erscheint von außen nach wie vor als ein Class B Netz.

| netid (16 Bit) | hostid (16 Bit) | |
|---|---|---|
| | | |
| netid (16 Bit) | subnetid (3 Bit) | hostid (13 Bit) |

Die benötigte Netzmaske muss die 3 höchstwertigen Bit des 3. Octet und die 16 Bit der netid ausmaskieren: Netzmaske = 255.255.224.0

Damit ist das Class B-Netz in 8 gleich große Subnetze unterteilt worden:

| Subnetz Nr. | niedrigste Adresse | höchste Adresse |
|---|---|---|
| 0 | 134.108.0.0 | 134.108.31.255 |
| 1 | 134.108.32.0 | 134.108.63.255 |
| 2 | 134.108.64.0 | 134.108.95.255 |
| 3 | 134.108.96.0 | 134.108.127.255 |
| 4 | 134.108.128.0 | 134.108.159.255 |
| 5 | 134.108.160.0 | 134.108.191.255 |
| 6 | 134.108.192.0 | 134.108.223.255 |
| 7 | 134.108.224.0 | 134.108.255.255 |

Zu beachten ist dabei, dass es nicht empfohlen wird, das Subnetz 0 und 7 zu verwenden, da die Netzadresse des Subnetzes 0 (134.108.0.0) identisch ist mit der Netzadresse des Class B-Netzes. Ebenso ist die Broadcastadresse des Subnetzes 7 identisch mit der Broadcastadresse des gesamten Class B-Netzes. Es verbleiben demnach 6 nutzbare Subnetze.

Auch innerhalb eines Subnetzes gibt es spezielle IP-Adressen, die nicht an einen Rechner vergeben werden dürfen:

- die niedrigste IP-Adresse eines Subnetzes, also die Adresse, bei der alle Bits der hostid = “0“sind (im Beispiel die Adressen 32.0, 64.0 usw.) ist die Netzadresse des gesamten Subnetzes
- die höchste IP-Adressen eines Subnetzes, also die Adresse, bei der alle Bits der hostid = “1“ sind (63.255, 95.255 usw.) ist die Broadcastadresse des Subnetzes

Um bei einer IP-Adresse deutlich zu machen, welcher Teil der Adresse das Netz bezeichnet, kann die Anzahl der für die Netzmaske verwendeten Bits hinter der IP-Adresse, durch einen Schrägstrich getrennt, angegeben werden. Im obigen Beispiel würde dann eine Adresse geschrieben als 134.108.21.44 / 19

#### 4.4.1.3 Aufbau eines IP-Paketes

Ebenso wie ein MAC-Frame besteht auch ein IP-Paket aus den eigentlichen Nutzdaten und aus Zusatzinformationen, die im Header des Pakets untergebracht sind. Bild 4-8 zeigt die im RFC 791 festgelegte Struktur des IP-Headers.

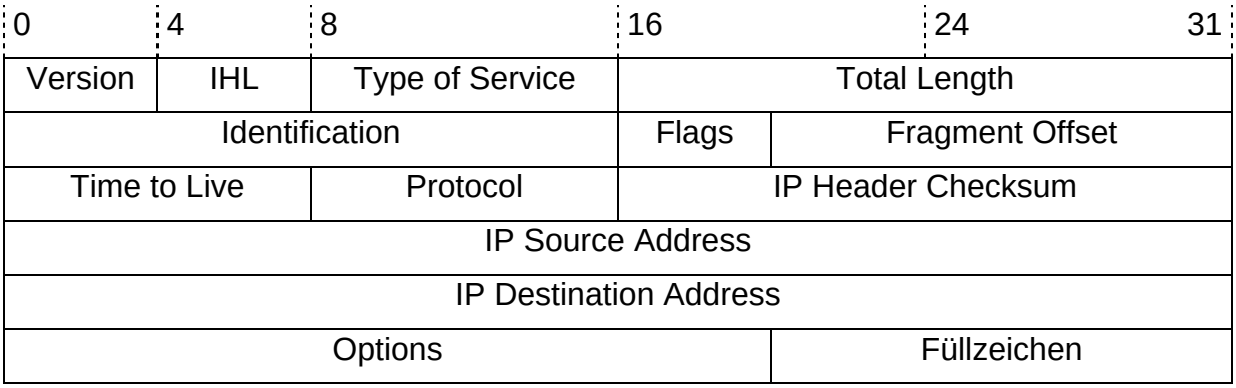



*Bild 4-8 Aufbau des IP-Headers*

In den nachfolgenden Abschnitten werden die wichtigsten Feldelemente des IP-Headers genauer erklärt [ip2, ip3, ip4].

**Version**

Abkürzung: Vers

Feldlänge: 4 Bit

Das Versions-Feld gibt die verwendete Version des IP-Protokolls an. In der Regel wird heute noch die Version 4 verwendet, die Standardisierung der Version 6 ist jedoch abgeschlossen

**Internet Header Length (LÄNGE)**
Abkürzung: IHL
Feldlänge: 4 Bit
Einheiten: 4 Octet-Gruppen
Bereich: 5-15 (Standardwert: 5)

Die Internet Header Length steht für die gesamte Länge des IP-Headers, ausgedrückt in 32-Bit-Einheiten (4-Octets). Das Internet Header Length-Feld ist durch die variable Länge des Optionen-Felds im IP-Header erforderlich, der am häufigsten verwendete Header hat eine Länge von 20 Octets, das IHL-Feld enthält dann also den Wert 5

**Type of Service (Servicetypen)**
Abkürzung: TOS, Service Typ

Feldlänge: 8 Bit

Dieses Feld gibt die gewünschte Qualität des Dienstes (Vorrang, Verzögerung, Durchsatz und Zuverlässigkeit) für dieses Datagramm an.

Die angesprochenen Netzknoten können entweder den gewünschten Dienst erbringen (imstande sein) oder leisten diese Dienste oder Teile davon nicht (nicht imstande sein).

**Total Length (Paketlänge)**
Abkürzung: TL
Feldlänge: 16 Bit
minimaler Wert: 20

Dieses Feld gibt die Länge des Datagramms (sowohl Kopf als auch Benutzerdaten), gemessen in Octets an. Da dieses Feld 16 Bit lang ist, kann ein IP-Paket inklusive Header maximal $2^{16}$ oder 65535 Octets lang sein.

**Identification (Identifikation)**
Abkürzung: ID
Feldlänge: 16 Bit

Dieses Feld enthält eine eindeutige Identifikation des IP-Pakets, z.B. einen Zähler, der durch den absendenden Host vergeben wird. Dieses Feld wird bei der Reassemblierung von Fragmenten verwendet, um alle Teile einer Fragmentkette identifizieren zu können.

**Flags (Flaggen)**
Abkürzung: keine
Feldlänge: 3 Bit

Zwei Bits namens DF ("don't fragment") und MF ("more fragment") steuern die Behandlung des Pakets im Falle der Fragmentierung. Ist das DF-Bit gesetzt, darf das IP-Paket unter keinen Umständen fragmentiert werden, auch wenn es dann nicht mehr weitertransportiert werden kann und verworfen werden muss. Das MF-Bit zeigt an, ob dem IP-Paket weitere Teilpakete nachfolgen. Das erste Bit dieses Feldes ist ungenutzt.

**Fragment Offset (Fragmentabstand)**
Abkürzung: FO
Feldlänge: 13 Bit
Einheiten: 8 Octet-Gruppen
Bereich: 0-8191 (Standardwert: 0)

Dieses Feld gibt die Lage der Fragmentdaten relativ zum Anfang des Datenblocks im ursprünglichen Datagramm an. Bei einem nicht fragmentierten Datagramm oder beim ersten Fragment ist der Wert des FO immer auf Null gesetzt.

Der FO definiert die Lage des jeweiligen Fragments als ein Vielfaches von 8 Byte (Grundeinheit der Fragmentierung). Durch die zur Verfügung stehenden 13 Bits sind maximal 8192 Fragmente pro Datagramm möglich (65536 Byte).

Das FO-Feld ermöglicht dem Empfänger, mehrere Fragmente in der richtigen Reihenfolge zusammenzusetzen.

**Time to Live (Lebenszeit)**
Abkürzung: TTL
Feldlänge: 8 Bit
Einheit: Sekunden
Bereich: 0-255

Der absendende Host gibt hier an, wie lange das Paket im Netz verweilen darf, bevor es weggeworfen werden muss. Jedes Mal, wenn das Datagramm in einem Netzkoppelelement die Internetschicht durchläuft, muss die IP-Einheit dieses Feld mindestens um eins vermindern. Somit ist die Lebenszeit meist gleichbedeutend mit der Anzahl der Netzknoten, die von einem Paket maximal durchlaufen werden können (= hops). Wenn dieses Feld den Wert 0 enthält, muss das Paket vom verarbeitenden Rechner weggeworfen werden. Somit wird verhindert, dass ein Paket endlos im Netz zirkuliert. Der Absender des Pakets erhält in diesem Fall eine ICMP-Nachricht über den Vorgang.

**Protocol (Protokoll)**
Abkürzung: PROT
Feldlänge: 8 Bit

Dieses Feld enthält die Identifikation des Transportprotokolls, dem das Paket zugestellt werden muss.
Die wichtigsten Protokolltypen sind:

1 = ICMP Internet Control Message Protocol

6 = TCP Transmission Control Protocol

17 = UDP User Datagramm Protocol

**IP Header Checksum (Kopfprüfsumme)**
Abkürzung: keine
Feldlänge: 16 Bit

Enthält eine Prüfsumme, die nur den IP-Header gegen Fehler sichert.
Mit der Prüfsumme für den IP-Header kann allerdings nur der IP-Header auf Fehler überprüft werden. Beim Durchgang durch einen Router verändert sich der Header (z.B. durch Herabsetzen des Wertes im TTL-Feld um Eins) und die Prüfsumme muss neu berechnet werden.

**IP Source Address (IP-Sendeadresse)**

Abkürzung: Source

Feldlänge: 32 Bit

Enthält die Internet-Adresse des Netzknotens, der das Datagramm erzeugt hat.

**IP Destination Address (IP-Empfängeradresse)**
Abkürzung: Dest
Feldlänge: 32 Bit

Enthält die Internet-Adresse des Netzknotens, für den das Datagramm bestimmt ist.

**Options (Optionen)**
Abkürzung: Opt
Feldlänge: variabel

Die Dienste (Netzmanagement, Sicherheit), die IP den speziellen Anforderungen eines höheren Protokolls anpasst, werden durch die Optionen definiert.

## 4.4.2 Übertragung von IP-Paketen

Um Daten mit einem Kommunikationspartner austauschen zu können, benötigt der Protokollstapel eines Rechners außer der IP-Adresse des Partners noch einige weitere Informationen:

- seine eigene MAC-Adresse, die entweder im Netzadapter hardwaremäßig codiert ist oder durch Jumper auf der Netzadapterkarte vergeben wird
- seine eigene IP-Adresse
- die Subnetzmaske
- die IP-Adresse eines Netzknotens, an den er alle Pakete schickt, für die er in seiner eigenen Wegetabelle keinen Weg eingetragen hat: das Default Gateway bzw. der Default Router

Diese Informationen (außer der MAC-Adresse) stehen üblicherweise in Konfigurationsdateien des Betriebssystems. Unter UNIX können einige dieser Parameter mit dem Befehl ifconfig kontrolliert bzw. verändert werden, bei den aktuellen 32-Bit Windows-Varianten (Windows NT, Windows 95) ist dies über die Netzwerkeigenschaften möglich. Bei plattenlosen Systemen können die letzten drei Informationen auch über ein spezielles Protokoll, das BOOTP-Protokoll, von einem Server bezogen werden.

### 4.4.2.1 Fragmentierung

Um die IP-Pakete über das darunterliegende Netz zu transportieren, müssen die IP-Pakete in MAC-Rahmen verpackt werden. Im Idealfall passt ein IP-Paket genau in den Datenteil eines MAC-Rahmens. Unglücklicherweise hat aber jede Netztechnologie eine unterschiedliche Obergrenze für die Menge an Daten, die in einem MAC-Rahmen transportiert werden kann. Diese Anzahl, Maximum Transfer Unit (MTU) genannt, beträgt z.B. bei Ethernet 1500 Octets , bei FDDI 4470 Octets, bei X.25 jedoch nur 128 Octets pro MAC-Rahmen.

Eine Möglichkeit zur Lösung des Problems wäre, die Größe eines IP-Paketes auf die kleinste mögliche MTU festzulegen, dies wäre jedoch reichlich ineffizient. Die Kommunikationspartner können auch nicht vor der Übertragung die Größe des IP-Paketes festlegen, da IP einen verbindungslosen Transportdienst bietet und jedes Paket einen anderen Weg durch das Netz nehmen kann. Es muss also innerhalb von IP eine Möglichkeit geben, die Größe der Datagramme flexibel der MTU des jeweiligen Netzes anzupassen; diese Möglichkeit ist die Fragmentierung.

Der absendende Knoten generiert Pakete, die auf die MTU des Netzes, an das er angeschlossen ist, optimiert sind. Soll dieses Paket dann in ein Netz übertragen werden, das eine kleinere MTU hat, wird das Paket in mehrere Teile, sogenannten Fragmente, zerteilt. Jedes dieser Fragmente wird in einen Rahmen des Netzes verpackt und weitertransportiert. Beim Empfänger müssen dann diese Fragmente wieder zusammengesetzt werden. Bild 4-9 zeigt an einem Beispiel, wie im Laufe eines Datentransfers trotz identischer MTU im Quell- und Zielnetz eine Fragmentierung stattfindet. In diesem Beispiel sind Host 1 und Host 2 an ein Ethernet angeschlossen und können Pakete mit einer Maximalgröße von 1500 Octets erzeugen. Der Pfad zwischen den beiden Rechnern enthält jedoch ein Netz

mit einer MTU von 128. Wenn Host 1 ein Paket an Host 2 schickt, dass größer als 128 Octets ist, so muss der Router $R_1$ das Paket in Fragmente mit einer Größe von 128 Octets zerlegen.

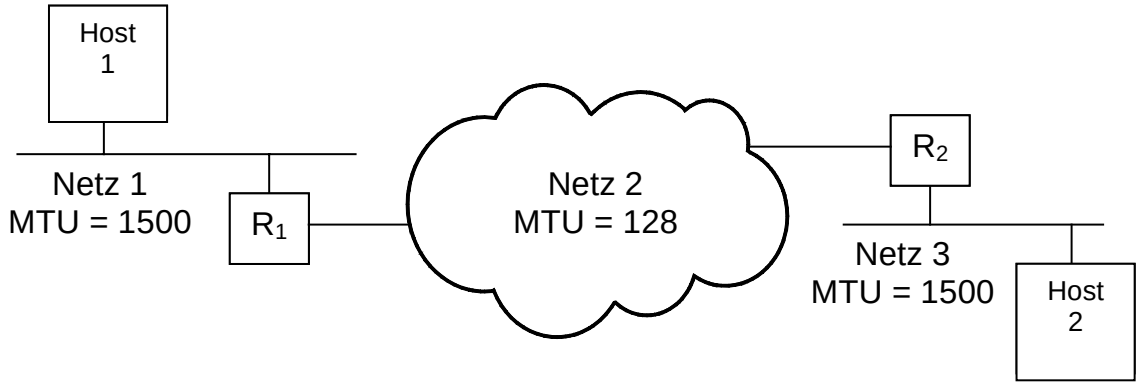



*Bild 4-9 Fragmentierung im Laufe eines Datentransfers*

Jedes Fragment einer Nachricht erhält einen vollständigen IP-Protokollkopf sowie das Identifikationsfeld der Ausgangs-Nachricht, mit deren Hilfe alle Fragmente einer Nachricht wiedererkannt werden. Die Lage der Daten eines Fragments innerhalb der Gesamtnachricht wird mit Hilfe des Fragmentabstandsfeld (Fragment Offset) ermittelt.

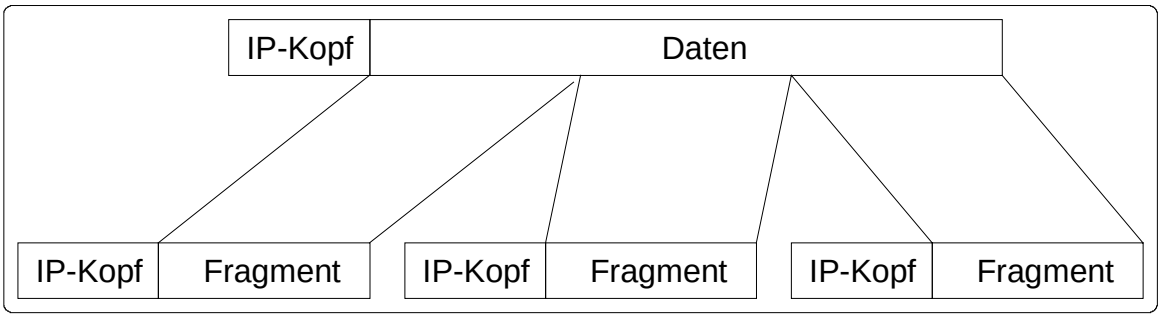



*Bild 4-10 Fragmentierung eines IP-Paketes*

Bild 4-10 zeigt die schematische Darstellung eines fragmentierten IP-Pakets. Eine etwas genauere Betrachtung zeigt Bild 4-11 [22]:

| Datagramm Header | data (1400 Octets) |
|---|---|

| Fragment 1 Header | $data_1$ (600 Octets) | Fragment 1 Offset 0 |
|---|---|---|
| Fragment 2 Header | $data_2$ (600 Octets) | Fragment 2 Offset 600 |
| Fragment 3 Header | $data_3$ (200 Octets) | Fragment 3 Offset 1200 |

*Bild 4-11 Beispiel für die Fragmentierung*

Zu sehen ist eine Fragmentierung, bei der ein Paket, dass ursprünglich 1400 Octets Nutzdaten enthielt, so fragmentiert wird, dass es über ein Netz mit einer MTU von 620 transportiert werden kann. Die Fragment Header sind dabei identisch mit dem Datagramm Header; lediglich das „more fragment“ Flag, das bei den Fragmenten 1 und 2 gesetzt ist, und das Fragment Offset-Feld, das die Lage im ursprünglichen Paket angibt, enthalten unterschiedliche Werte.

Wenn der absendende Rechner verhindern will, dass das Paket fragmentiert wird, etwa weil der Empfänger nicht zur Reassemblierung in der Lage ist, kann er die Fragmentierung durch setzen des "Don't Fragment"-Flag im IP-Header verhindern. Die Nachricht wird dann verworfen, wenn sie über ein Netz nicht ohne Fragmentierung übertragen werden kann.

Das ursprüngliche Paket wird erst beim Empfänger der Nachricht zusammengesetzt, selbst wenn die Fragmente weitere Netze mit größerer MTU durchlaufen. Dies ist schon deswegen notwendig, weil nicht alle Fragmente den gleichen Weg zum Empfänger nehmen müssen. Sobald das erste Fragment einer Nachricht beim Empfänger eintrifft, startet der Empfänger einen Timer, den *reassembly timer*. Läuft dieser Timer ab, bevor alle restlichen Fragmente eingetroffen sind, wird die noch unvollständige Nachricht weggeworfen. Auf diese Weise wird verhindert, dass unvollständige Nachrichten für immer unnötigen Pufferspeicher belegen, wenn ein Fragment der Nachricht verloren geht.

Bei der Reassemblierung werden die Felder Identification, Fragment Offset und das MF-Flag des IP-Headers benötigt. Durch das Feld Identification kann der Empfänger eindeutig feststellen, zu welchem Paket das Fragment gehört. Der Wert des Fragment Offset Feldes gibt die Lage des Fragmentes innerhalb des gesamten Pakets an und mit Hilfe des MF-Flags kann festgestellt werden, ob das aktuelle Fragment das letzte Fragment des Paketes ist.

### 4.4.3 ARP und RARP

ARP über Ethernet definiert in RFC 826 (Nov. 1982)

RARP definiert in RFC 903 (Juni 1984)

Bisher haben wir uns mit der Feststellung zufriedengegeben, dass mit Hilfe der IP-Adresse der Zielrechner adressiert wird. Um einen Rechner tatsächlich zu adressieren, wird jedoch nicht die IP-Adresse benutzt, sondern die Hardware-Adresse (MAC-Adresse) des Rechners. Es muss also möglich sein, die IP-Adresse in eine MAC-Adresse umzusetzen und damit das Paket zum Partner zu übertragen.

Grundsätzlich gäbe es für die Umsetzung von einer Internet-Adresse in eine MAC-Adresse (z.B. Ethernet) drei Möglichkeiten [22]:

1. **Statische Umsetzung über eine Tabelle**
   Dies hat den Nachteil, dass bei Veränderung im Netz ein Neuladen der Tabelle erfolgen muss und diese Tabelle von Hand gepflegt werden muss.
2. **Umwandeln der Internet- in eine MAC-Adresse mit Hilfe einer Formel**
   Wollte man eine MAC-Adresse mit Hilfe einer Formel in die IP-Adresse umwandeln, so würde dies bedeuten, dass beispielsweise einer Station mit der Ethernet-Adresse 80-04-00-60-00-01 die Internet-Adresse 89.60.00.01 zugeordnet werden müsste, d.h. Teile der Ethernet-Adresse würden für die Internet-Adresse verwendet. Allerdings besteht die Möglichkeit, dass eine solche Umsetzung nicht eindeutig ist: wenn die für die Umsetzung verwendeten Felder zweier Ethernet-Adressen zufällig gleich sind, dann besitzen zwei Hosts dieselbe Internet-Adresse. Diese Möglichkeit ist nur bei Netzen sinnvoll, bei denen die Hardware-Adresse vom Benutzer wählbar ist, z.B. über Jumper auf der Netzadapterkarte.
3. **Dynamische Umsetzung durch Abfrage im Subnetzwerk**
   Dadurch werden z.B. Veränderungen der Ethernet-Adresse im Netz transparent. Diese Möglichkeit ist nur auf einem LAN mit Broadcast sinnvoll durchzuführen. Zu diesem Zweck wurde das *Address Resolution Protocol* (ARP) geschaffen, mit dessen Hilfe die Umsetzung der IP-Adresse in die MAC-Adresse, z.B. in die 48 Bit MAC-Adresse des Ethernet, durchgeführt wird.

Die Idee des ARP ist einfach: wenn Rechner A eine Nachricht an Rechner B schicken möchte, sendet er vorher ein spezielles Paket als MAC-Broadcast aus, das nach der zur IP-Adresse gehörenden MAC-Adresse fragt (der sogenannte *ARP-Request*). Der Rechner, der die gesuchte IP-Adresse hat (also Rechner B), beantwortet das Paket mit einem *ARP-Reply*. Nun kann Rechner A die Nachricht an Rechner B schicken. Damit nicht vor jeder Übertragung ein ARP-Request ausgesendet werden muss, wird die Zuordnung der IP-Adresse zur MAC-Adresse in einem Cache, dem *ARP-Cache*, zwischengespeichert. Wenn eine Nachricht verschickt werden soll, wird zunächst überprüft, ob der ARP-Cache einen Eintrag für die benötigte IP-Adresse enthält. Ist dies der Fall, so muss nicht zuerst die MAC-Adresse mit einem Broadcast ermittelt werden, sondern die Übertragung kann mit der Adresse aus dem ARP-Cache erfolgen.

Damit diese Methode möglichst effizient funktioniert, können einige Tricks benutzt werden. So kann beispielsweise Rechner B, wenn er einen ARP-Request von Rechner A empfängt, die Zuordnung von Rechner A's IP-Adresse zu dessen MAC-

Adresse, die er dem ARP-Paket entnehmen kann, in seinem ARP-Cache abspeichern. Ebenso können alle anderen Rechner im Netz verfahren, die diesen Broadcast empfangen. Damit wäre bei einer Kommunikation mit A kein neuerlicher Broadcast mehr nötig.

Natürlich kann der Eintrag im ARP-Cache nicht auf ewig bestehen bleiben, da sich die MAC-Adresse eines Rechners, beispielsweise durch den Austausch einer Netzadapterkarte, ändern kann. Der Eintrag wird deshalb nach einer festgelegten, implementationsabhängigen Zeit (meist zwischen einer und zwanzig Minuten) aus dem Cache gelöscht, wobei die Zeit bei jedem empfangenen ARP-Paket, das die Zuordnung bestätigt, zurückgesetzt wird.

Zur Verdeutlichung zeigt Bild 4-12 den Vorgang eines ARP-Requests:

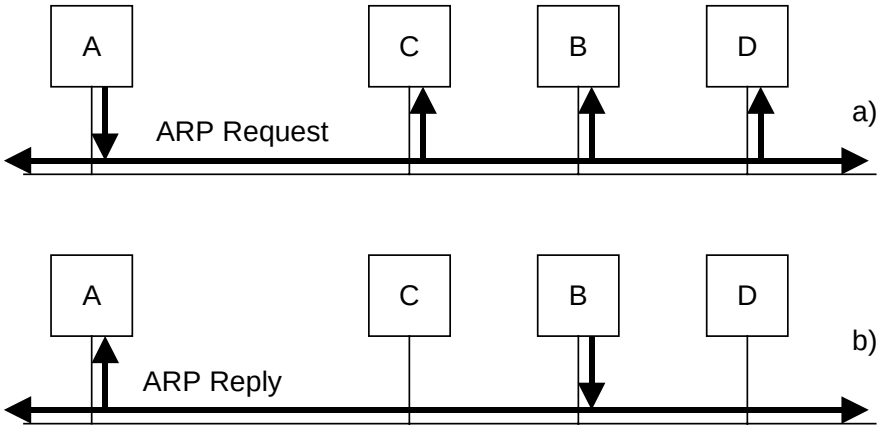



*Bild 4-12 Ablauf eines ARP-Requests*

Rechner A sendet zunächst einen ARP-Request als MAC-Broadcast aus (Bild 4-12a). Alle Rechner innerhalb des physikalischen Netzsegmentes empfangen diesen Broadcast und tragen die Zuordnung der IP-Adresse von Rechner A zu dessen MAC-Adresse, die sie aus dem ARP-Request entnehmen konnten, in ihren ARP Cache ein. Nur der Rechner, nach dessen MAC-Adresse gefragt wurde (Rechner B) beantwortet den Request mit einem ARP-Reply (Bild 4-12b), der nicht als Broadcast, sondern direkt zurück an Rechner A geschickt wird. Nun weiß Rechner A die MAC Adresse von Rechner B und kann die Nachricht an Rechner B versenden.

ARP-Requests werden, wie alle Pakete, als Daten in einem MAC-Rahmen übertragen. Um den Rahmen zu kennzeichnen, wird das Type-Feld im MAC-Header auf einen entsprechenden Wert gesetzt, im Falle von Ethernet hat dieses Feld bei einer ARP-Nachricht den Wert 0806h.

Im Gegensatz zu den meisten anderen Protokollen sind die Längen der Felder im ARP-Header nicht festgelegt. Dies ist nötig, damit ARP auf jeder erdenklichen Netzhardware arbeitet. Stattdessen enthält der Header zu Beginn einige Felder, die unter anderem festlegen, welche Länge die variablen Felder haben.

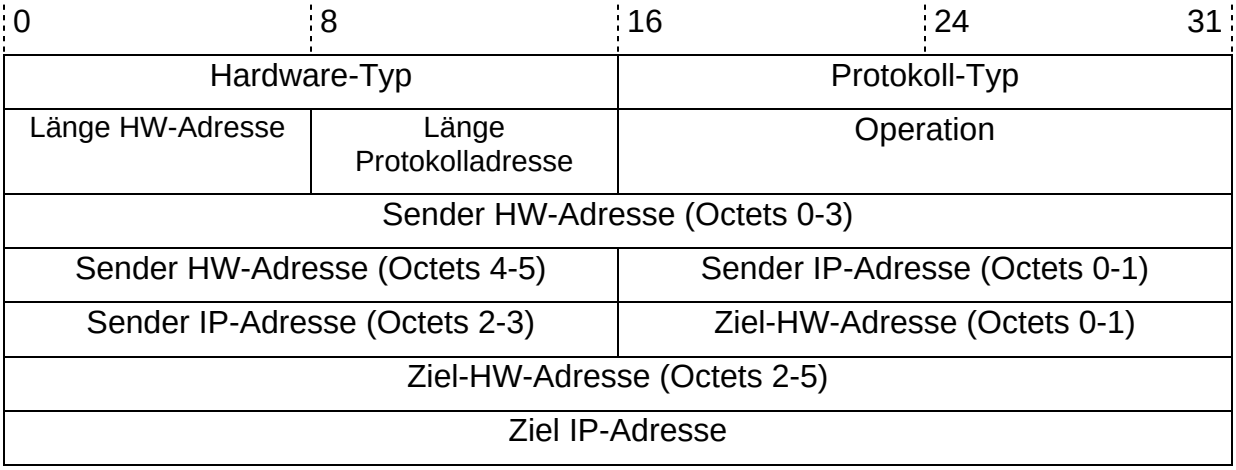



*Bild 4-13 Aufbau einer ARP-Nachricht im Ethernet*

Die Felder haben die folgende Bedeutung [24]:

**Hardware-Typ**
Abkürzung: HRD
Feldlänge: 16 Bit

Definiert das verwendete Übertragungsmedium, Kommunikationsgeschwindigkeit und Datenstrukturen, z.B.

1 = Ethernet

6 = IEEE 802.x Netzwerke

**Protokoll-Typ**
Abkürzung: PRO
Feldlänge: 16 Bit

Typ-Feld-Nummern zur Unterscheidung des verwendeten Netzprotokolls, 0800h für IP.

**Länge HW-Adresse**
Abkürzung: keine
Feldlänge: 8 Bit

Anzahl der Hardware-Adress-Octets. Bei Ethernet beträgt der Wert immer "06", was einer Länge von 48 Bit entspricht.

**Länge Protokoll**
Abkürzung: keine
Feldlänge: 8 Bit

Anzahl der Adress- Octets die durch höhere Protokolle (z.B. IP) definiert werden. Bei TCP/IP-Protokollen beträgt der Wert immer "04".

**Operation**
Abkürzung: OP
Feldlänge: 16 Bit

Definiert die Art des ARP-Paketes:
1 = ARP-Anfrage (Request)
2 = ARP-Antwort (Reply)

3 = RARP-Anfrage (Request)

4 = RARP- Antwort (Reply)

**Sender HW-Adresse**
Abkürzung: Source
Feldlänge: 48 Bit

Die Sender HW-Adresse gibt die Hardware-Adresse des Senders an.

**Sender IP-Adresse**
Abkürzung: So-Adr
Feldlänge: 48 Bit

IP-Adresse des Senders.

**Ziel HW-Adresse**
Abkürzung: De-Adr
Feldlänge: 48 Bit

Dieses Feld enthält bei einem ARP-Reply die Hardware Adresse des Empfängers. Bei einem ARP-Request ist die Hardware-Adresse des Zielknotens nicht bekannt, deshalb wird bei ARP-Anfragen dieses Feld mit der Broadcast-Adresse des Netzes belegt. Diese Adresse wird bei der Antwort durch die richtige Hardware-Adresse ersetzt.

**Ziel IP-Adresse**
Abkürzung: Pro-Dest
Feldlänge: 32 Bit

IP-Adresse des Empfängers.

Der Aufbau eines Paketes des Reverse Address Resolution Protocol (RARP) ist identisch zum Aufbau eines ARP-Paketes. Im Gegensatz zu ARP ermittelt RARP zu einer bekannten MAC-Adresse eine IP-Adresse. Dies ist notwendig, wenn z.B. eine Host ohne eigene Festplatte gebootet wird. Da die Konfigurationsdateien zusammen mit dem Betriebssystem des Hosts auf einem Server abgelegt sind, muss der Host eine Möglichkeit haben, seine eigene IP-Adresse zu erfahren, um sein Betriebssystem vom Server zu laden. Dazu versendet er einen RARP-Request als MAC-Broadcast, das im Typ/Längenfeld den Wert 0835h enthält. Der RARP-Server empfängt dieses Paket, sucht in seiner Tabelle nach der zur MAC-Adresse gehörenden IP-Adresse und sendet den RARP-Reply direkt an den Host. Dieser kann nun über TCP/IP sein Betriebssystem vom Server laden.

## 4.4.4 Beispiele für die Übertragung eines IP-Paketes

Um die bis hier besprochenen Sachverhalte zu vertiefen, werden nun an einigen Beispielen die Vorgänge bei der Übertragung eines IP-Paketes erläutert. Es wird davon ausgegangen, dass vor diesen Beispielen kein Netzverkehr stattgefunden hat und daher alle Caches leer sind.

Den Beispielen liegt die folgende Netzstruktur zugrunde:

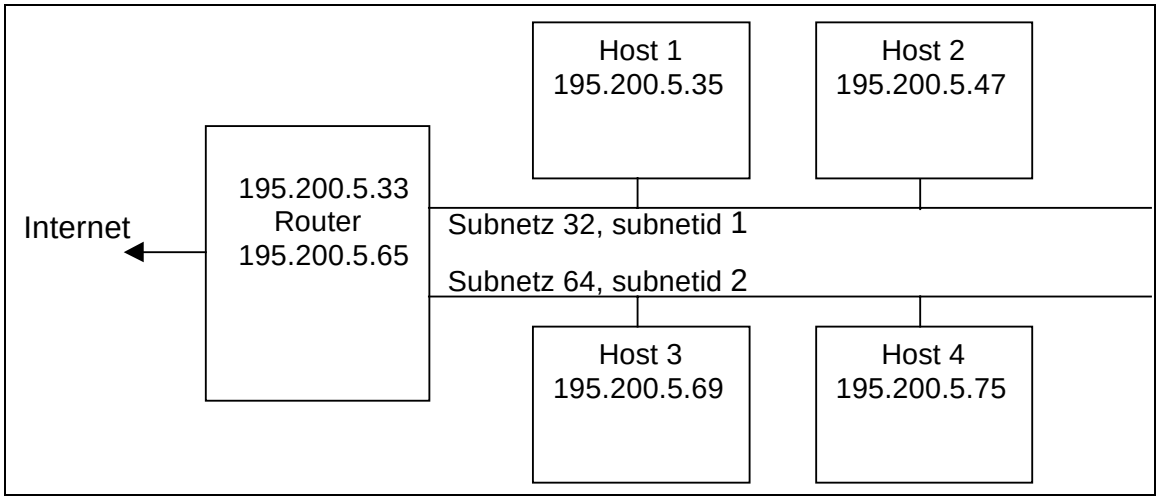



*Bild 4-14 Ethernet-LAN mit 2 IP-Subnetzen*

Es handelt sich hierbei um ein Class C-Netz, das in 8 Subnetze unterteilt wurde. Die hierzu benötigte Netzmaske lautet 255.255.255.224. Es ist allgemein üblich, dem Default Router des Netzes entweder die erste oder die letzte gültige Adresse des Subnetzes zuzuteilen – dies erleichtert das Merken der IP-Adresse. Im Beispiel werden die ersten beiden Subnetze benutzt: die Netze mit der Netznummer 195.200.5.32 (subnetid 1, da das niederwertigste Bit der subnetid gesetzt ist) und 195.200.5.64.

### 4.4.4.1 Übertragung eines IP-Paketes an einen Host im gleichen Subnetz

Als erstes betrachten wir den einfachsten Fall: Host 1 möchte ein Paket an Host 2 schicken, der sich in seinem Subnetz befindet. Um dies herauszufinden, muss jedoch zuerst die Netzadresse des Zielknotens mit der eigenen Netzadresse verglichen werden:

Host 1:

| | |
|---|---|
| IP-Adresse 1: | 11000011 11001000 00000101 00100011 |
| Netzmaske 1: | 11111111 11111111 11111111 11100000 |
| **Netzadresse 1:** | **11000011 11001000 00000101 00100000** |

Host 2:

| | |
|---|---|
| IP-Adresse 2: | 11000011 11001000 00000101 00101111 |
| Netzmaske 2: | 11111111 11111111 11111111 11100000 |
| **Netzadresse 2:** | **11000011 11001000 00000101 00100000** |

Aus der Gleichheit der beiden Netzadressen folgt, dass sich der Zielknoten im gleichen Subnetz befinden und das Paket direkt an diesen Knoten geschickt werden kann. Da die MAC-Adresse des Zielknotens sich nicht im ARP-Cache befindet, sendet Host 1 folgenden ARP-Request aus, um die MAC-Adresse von Host 2 herauszufinden:

MAC-DA = FFFFFFFFh (Broadcast)

MAC-SA = MAC-Adresse (Host 1)

Type = 0806h (ARP)

IP-DA = 195.200.5.47

IP-SA = 195.200.5.35

Operation = 1 (ARP-Request)

Da es sich bei diesem Paket um einen MAC-Broadcast handelt, empfangen alle Knoten im Subnetz dieses Paket. Der Router leitet diesen Broadcast jedoch nicht weiter, so dass Host 3 und 4 dieses Paket nicht empfangen.

Nur der Host, dessen IP-Adresse mit der angefragten IP-Adresse übereinstimmt, beantwortet den ARP-Request mit einem ARP-Reply. Dabei wird vorausgesetzt, dass eine IP-Adresse nur einmal vorkommt, bei Adressduplikaten kommt es zu einem Fehler. Der ARP-Reply wird von Host 2 direkt an Host 1 adressiert und enthält die folgenden wesentlichen Felder:

MAC-DA = MAC-Adresse (Host 1)

MAC-SA = MAC-Adresse (Host 2)

Type = 0806h (ARP)

IP-DA = 195.200.5.35

IP-SA = 195.200.5.47

Operation = 2 (ARP-Reply)

Host 1 kennt nun die MAC-Adresse von Host 2 und kann das Datenpaket an Host 2 senden. Beide Hosts tragen die Zuordnung IP-Adresse ↔ MAC-Adresse in ihren Cache ein und können in Zukunft, solange der Eintrag gültig ist, ohne einen vorhergehenden ARP-Request miteinander verkehren.

#### 4.4.4.2 Übertragung eines IP-Paketes an einen Host im anderen Subnetz

Im zweiten Fall betrachten wir die Übertragung eines Pakets von Host 1 zu Host 3, der sich im anderen Subnetz befindet. Wie bei jeder Übertragung wird auch in diesem Fall zunächst die Netzadresse des Zielknotens mit der eigenen Netzadresse verglichen:

Host 1:

| | |
|---|---|
| IP-Adresse 1: | 11000011 11001000 00000101 00100011 |
| Netzmaske 1: | 11111111 11111111 11111111 11100000 |
| **Netzadresse 1:** | **11000011 11001000 00000101 00100000** |

Host 3:

| | |
|---|---|
| IP-Adresse 3: | 11000011 11001000 00000101 01000101 |
| Netzmaske 3: | 11111111 11111111 11111111 11100000 |
| **Netzadresse 3:** | **11000011 11001000 00000101 01000000** |

Nun ist die Netzadresse des Zielknotens nicht mehr mit der des Quellknotens identisch. Das Paket kann also nicht direkt an den Zielknoten geschickt werden, da sich dieser nicht im eigenen Subnetz befindet. Stattdessen muss das Paket über einen weiteren Rechner, den Default Router, an Knoten 3 geschickt werden. Der Default Router ist derjenige Rechner, an den ein Host alle Pakete schickt, die nicht an einen Host im eigenen Subnetz adressiert sind und für die er keinen Eintrag in seiner lokalen Routing-Tabelle hat. Da Host 1 die IP-Adresse des Default-Routers aus seinen Konfigurationsdaten kennt, aber noch nicht dessen MAC-Adresse, muss er zunächst die MAC-Adresse des Routers durch einen ARP-Request herausfinden. Ist dies geschehen, schickt Host 1 ein IP-Paket an den Router, das als MAC-DA die Adresse des Routers enthält, als IP-DA jedoch die IP-Adresse von Host 3. Die wesentlichen Felder im Rahmen enthalten folgende Werte:

MAC-DA = MAC-Adresse Router (Subnetz 32)

MAC-SA = MAC-Adresse Host 1

Type = 0800 (IP)

IP-DA = 195.200.5.69

IP-SA = 195.200.5.35

Der Router empfängt das IP-Paket und entscheidet anhand der IP-Adresse, an welchen seiner Anschlüsse er das Paket weiterleitet. Durch einen Vergleich der IP-Adresse mit den Einträgen in seiner Routing-Tabelle erkennt er, dass an einem anderen Anschluss ein „passendes“ Netz angeschlossen ist. Er leitet das Paket in das Subnetz weiter, muss sich aber zunächst über ARP die MAC-Adresse des Hosts mit der im IP-Paket stehenden IP-Adresse beschaffen. Erst dann kann er das Paket an Host 3 senden, wobei als MAC-SA seine MAC-Adresse im entsprechenden Rahmenfeld steht:

MAC-DA = MAC-Adresse Host 3

MAC-SA = MAC-Adresse Router (Subnetz 64)

Type = 0800 (IP)

IP-DA = 195.200.5.69

IP-SA = 195.200.5.35

Möchte nun z.B. Host 4 ein Paket an Host 1 senden, sind keinerlei ARP-Pakete mehr nötig, da Host 4 durch mithören des ARP-Requests im gerade besprochenen Fall (Router → Host 3) die MAC-Adresse des Routers kennt. Der Router kennt die MAC-Adresse von Host 1 bereits aus dem ersten Fall durch mithören des ARP-Requests von Host 1 an Host 2.

#### 4.4.4.3 Übertragung eines IP-Paketes an einen Host außerhalb des eigenen Netzes

Als letzten Fall betrachten wir noch die Situation, dass Knoten 1 ein Paket an einen Host schickt, der außerhalb des lokalen Netzes liegt. Host 1 vergleicht zunächst die Netzadresse des Zielknoten mit seiner Netzadresse und stellt fest, dass der Host außerhalb seines Subnetzes liegt. Daher sendet er das Paket an den Router. Dieser stellt durch überprüfen seiner Routing-Tabelle fest, dass sich der Host nicht in einem der an seinen Anschlüssen befindlichen Netze befindet. Da er für diesen Host auch keinen speziellen Eintrag in seiner Routing-Tabelle findet, schickt er das Paket an seinen Default-Router im Internet. Die weitere Vermittlung erfolgt durch die Router im Internet.

### 4.4.5 ICMP – Internet Control Message Protocol

Definiert in RFC 792 (Sept. 1981)

Das **Internet Control Message Protocol (ICMP)** ist ein Protokoll der Internet-Schicht und erlaubt es einer IP-Software auf einem Rechner, an die IP-Software eines anderen Rechners Kontroll- oder Fehlermeldungen zu schicken. Diese Möglichkeit wurde geschaffen, damit Router den Hosts den Grund eines Fehlers bei der Zustellung eines IP-Paketes zustellen können. ICMP ist Bestandteil jeder IP-Implementierung und transportiert Fehler- und Diagnoseinformationen für IP. Das ICMP-Paket wird im Datenteil eines IP-Paketes transportiert, seine Transportprotokoll-Adresse im Protokoll-Feld des IP-Headers ist "1". Dennoch wird ICMP nicht als ein Protokoll der höheren Schicht, sondern als Bestandteil der Internet-Schicht betrachtet.

Jede ICMP-Nachricht hat ihr eigenes Format, alle beginnen jedoch mit drei identischen Feldern:

- einem 8 Bit TYPE-Feld, das die Art der ICMP-Nachricht angibt
- einem 8 Bit CODE-Feld, das weitergehende Informationen enthalten kann
- und einem 16 Bit CHECKSUM-Feld, das eine Prüfsumme über das ICMP-Paket enthält.

Die restlichen Felder des Paketes sind abhängig von der ICMP-Nachricht, als Beispiel hier das Format einer Destination Unreachable-Nachricht [22]:

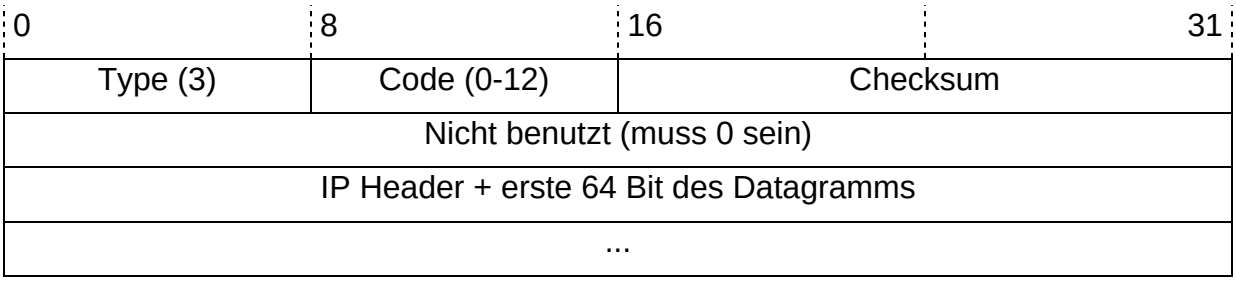



*Bild 4-15 Format einer Destination Unreachable-Nachricht*

Mit diesem Datagramm wird dem Absender mitgeteilt, dass sein Paket nicht zugestellt werden konnte. Das CODE-Feld enthält weitergehende Informationen zum Grund des Fehlers, wie z.B. Network Unreachable (0) oder Fragmentation needed and DF set (4). Im Datenteil sind außerdem der Header des betroffenen Paketes und die ersten 64 Bit des Datagramms enthalten, woraus der Sender zweifelsfrei ermitteln kann, welches Paket gemeint ist.

Die von ICMP unterstützten Kontrollnachrichten sind [22, 23]:

**destination unreachable:** Type 3
Ein Datagramm konnte nicht zugestellt werden, da ein Netzwerk oder Rechner nicht erreichbar war, ein Protokoll nicht betriebsbereit war, oder Fragmentierung notwendig gewesen wäre, aber durch das Flag-Feld (im IP-Header) verboten wurde.

**time exceeded:** Type 11
Ein Datagramm wurde weggeworfen, da seine Lebensdauer ablief oder ein Fragment wurde weggeworfen, weil es zu lange in der Warteschlange für die Reassemblierung war.

**parameter problem:** Type 12
Der Absender eines IP-Datagramms wird verständigt, dass das Paket aufgrund von fehlerhaften Angaben im IP-Protokollkopf weggeworfen werden musste.

**source quench:** Type 4
Ein Netzwerkgerät wirft Datagramme weg, da es zuwenig Betriebsmittel (z.B. Zwischenspeicher) hat.

**redirect:** Type 5
Wird ausgesendet, wenn ein Gateway erkennt, dass der Absender eines IP-Pakets dieses direkt an den nächsten Gateway senden könnte, d.h. ein unnötiger Umweg gegangen wird. Die ICMP-Nachricht enthält die Internet-Adresse des nächsten direkten Gateways.

| | |
|---|---|
| **echo request / echo reply:** Type 8 / 1 | Zum Test, ob eine IP-Adresse existiert, wird eine *echo request*-Nachricht gesendet. Nach Erhalt einer solchen Nachricht antwortet die empfangende IP-Einheit mit einer *echo reply*-Nachricht. |
| **timestamp request/ timestamp reply:** Type 13 / 14 | Zum Feststellen der Verzögerung im Netzwerk zwischen zwei Netzwerkgeräten. |
| **information request / information reply:** Type 15 / 16 | Zur Bestimmung der Adresse des eigenen IP-Netzwerks (wird inzwischen als nicht mehr notwendig betrachtet und vom Gebrauch wird abgeraten) |
| **address mask request / address mask reply:** Type 17 / 18 | Zur Bestimmung der Subnetz-Adressmaske. |

#### 4.4.5.1 ping

Zwei bekannte Programme die ICMP-Nachrichten benutzen sind die Programme ping und traceroute.

Mit ping wird eine echo request-Nachricht an einen Host geschickt. Wird dieses Paket mit einer echo reply-Nachricht beantwortet, ist sichergestellt, dass der Host erreichbar ist. Damit ist gleichzeitig sichergestellt, dass der IP-Protokollstapel auf Sender- und Empfängerseite korrekt arbeitet und alle Router auf dem Weg zwischen Sender und Empfänger korrekte Einträge in ihren Routing-Tabellen haben. Daher stellt dieses Programm ein wichtiges Hilfsmittel beim Test von TCP/IP-Implementierungen dar.

Je nach Implementierungen des ping-Kommandos werden mehrere ICMP echo request Pakete ausgesendet und eine Statistik über die Anzahl der verloren-gegangenen Pakete und die Zeit, die die Pakete benötigten, angezeigt.

Beispiel:

```
rshx01:~> PING 134.108.56.60: 64 byte packets
64 bytes from 134.108.56.60: icmp_seq=0. time=1. ms
64 bytes from 134.108.56.60: icmp_seq=1. time=1. ms
64 bytes from 134.108.56.60: icmp_seq=2. time=1. ms
64 bytes from 134.108.56.60: icmp_seq=3. time=1. ms
64 bytes from 134.108.56.60: icmp_seq=4. time=1. ms
64 bytes from 134.108.56.60: icmp_seq=5. time=1. ms
^C
----134.108.56.60 PING Statistics----
6 packets transmitted, 6 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 1/1/1
```

#### 4.4.5.2 traceroute

Mit Hilfe des Programms traceroute kann der Weg, den ein IP-Datagramm nimmt, verfolgt und angezeigt werden. Dazu sendet traceroute eine Reihe von IP-Datagrammen aus, bei denen das TTL-Feld von 1 ansteigende Werte enthält. Aus den zurückkommenden TIME EXEEDED-Meldungen kann entnommen werden, welche Stationen das Datagramm durchlaufen hat.

Beispiel:

```
rshx01:~> traceroute 192.151.11.13
traceroute to www.hp.com (192.151.11.13), 30 hops max, 20 byte packets
 1 rscs10.rz.fht-esslingen.de (134.108.50.249)     2 ms    2 ms    2 ms
 2 rscsbw.rz.fht-esslingen.de (134.108.48.253)     6 ms    2 ms    2 ms
 3 Hohenheim1.BelWue.DE (129.143.1.217)            3 ms    5 ms    3 ms
 4 Uni-Hohenheim1.WiN-IP.DFN.DE (188.1.10.5)       6 ms    7 ms    6 ms
 5 ZR-Stuttgart1.WiN-IP.DFN.DE (188.1.10.1)        6 ms    7 ms    6 ms
 6 ZR-Karlsruhe1.WiN-IP.DFN.DE (188.1.144.41)      9 ms    7 ms    8 ms
 7 ZR-Frankfurt1.WiN-IP.DFN.DE (188.1.144.37)     14 ms   15 ms   14 ms
 8 IR-Perryman1.WiN-IP.DFN.DE (188.1.144.78)         120 ms  126 ms  128 ms
 9 bordercore3.Washington.mci.net (166.48.41.249)    115 ms 111 ms 109 ms
10 core5.Washington.mci.net (204.70.4.109)           115 ms  127 ms  122 ms
11 wtn-uunet2-nap.Washington.mci.net (206.157.77.30) 117 ms 117 ms 111 ms
12 112.ATM10-0-0.XR1.TCO1.ALTER.NET (146.188.160.82) 120 ms 122 ms 120 ms
13 193.ATM10-0-0.TR1.DCA1.ALTER.NET (146.188.161.170)123 ms 140 ms 111 ms
14 101.ATM8-0-0.TR1.SCL1.ALTER.NET (146.188.136.222) 181 ms 196 ms 181 ms
15 100.ATM8-0-0.XR1.SCL1.ALTER.NET (146.188.145.233) 176 ms 188 ms 192 ms
16 195.ATM10-0-0.GW2.PAO1.ALTER.NET (146.188.144.77) 186 ms 201 ms 192 ms
17 * * *
18 hpcc923.external.hp.com (192.151.11.13)           181 ms  187 ms  190 ms
```

## 4.5 Transport-Schicht

### 4.5.1 Adressierung auf Ebene der Transportschicht

Ein Betriebssystem ist in der Regel multitasking-fähig. Dies bedeutet, dass mehrere Programme parallel bzw. quasi-parallel ablaufen können. Diese Programme laufen in einem oder mehreren Prozessen bzw. Tasks ab.

Die Identifizierung eines Prozesses erfolgt durch eine eindeutige Prozessnummer, der sogenannten Prozess-ID. Diese Prozess-ID wird dynamisch beim Start des Prozesses durch das Betriebssystem erzeugt. Es ist möglich, dass ein Prozess bei jedem Neustart eine andere Prozess-ID besitzen kann.

Für eine Identifizierung des Prozesses als Ziel einer Nachricht ist die Prozess-ID denkbar ungünstig, da alle potentiellen Sender bei einem Neustart des Zielprozesses informiert werden müssten, wie die neue Prozess-ID lautet. Aus diesem Grund wurde eine vom Betriebssystem unabhängige Einheit geschaffen, der Protokoll-Port. Das Betriebssystem hat dafür zu sorgen, dass Prozesse zum Senden von Daten eine Portnummer benutzen und leitet beim Empfang von Daten diese dem entsprechenden Prozess zu.

Zur Wiederholung:

> Das IP-Protokoll regelt den Datenaustausch zwischen den Knoten des IP-Netzes. Mit Hilfe der IP-Adresse können also Rechner adressiert werden. Da auf einem Rechner mehrere Prozesse, d.h. mehrere Programme ablaufen können, muss eine zusätzliche Differenzierung, der Protokoll-Port, eingeführt werden.

Die meisten Betriebssysteme stellen einen synchronen Zugriff auf die Ports zur Verfügung. Synchron bedeutet, dass ein Prozess z.B. bei einem Lesezugriff auf einen Port solange geblockt wird, bis Daten für diesen Port empfangen werden. Sobald Daten empfangen werden, wird der Prozess wieder gestartet. Im allgemeinen sind Protokoll-Ports auch gepuffert, sodass Daten zwischengespeichert werden, wenn ein Prozess noch nicht lesebereit ist.

### 4.5.2 UDP – User Datagram Protocol

Definiert in RFC 768 (28. August 1980).

Das User Datagram Protocol ermöglicht einem Programm das Senden und Empfangen von Datagrammen zu/von anderen Programmen. UDP unterstützt dabei die Eigenschaft der Protokoll-Ports, mit denen verschiedene Programme auf einem Rechner unterschieden werden können. Eine UDP-Nachricht enthält dabei neben den Daten sowohl den Ziel- als auch den Sender-Port, um einerseits der UDP-Software zu ermöglichen das Ziel zu finden, andererseits auch dem Empfänger zu ermöglichen eine Antwort zu schicken.

UDP benutzt IP, um Nachrichten an andere Rechner zu transportieren. UDP basiert dabei auf demselben ungesicherten und verbindungslosen Datagramm-Service wie IP. Es werden keine Quittungen verschickt und eingehende Nachrichten werden auch nicht sortiert. Somit können UDP-Nachrichten verlorengehen, gedoppelt werden oder auch defekt ankommen.

### 4.5.2.1 Das Format einer UDP-Nachricht

Eine UDP-Nachricht bzw. ein UDP-Datagramm besteht aus 2 Teilen. Einem UDP-Header und den UDP-Daten (siehe Bild 4-16).

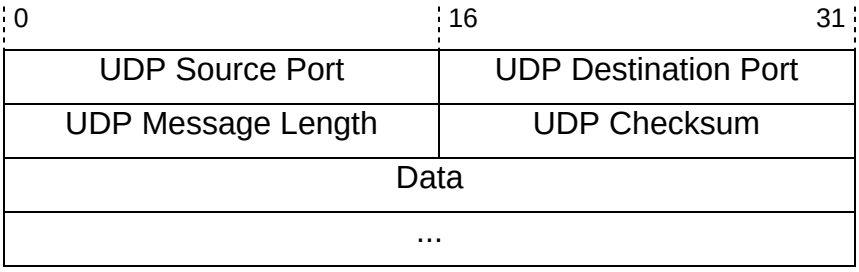



*Bild 4-16 Format einer UDP-Nachricht*

**Source Port, Destination Port:**

Source- und Destination-Port sind 16-Bit UDP Ports, wobei der Source-Port optional ist. Wenn er benutzt wird, so ist dies der Port, an den Antworten geschickt werden können, ansonsten sollte der Wert 0 sein.

**Length:**

Anzahl der Octets im UDP-Datagramm - UDP-Daten inklusive des UDP-Headers. Der Minimalwert des Inhalts ist 8 – die Länge des Headers ohne Daten.

**UDP Checksum:**

Die Prüfsumme ist optional. Der Wert 0 bedeutet, dass die Prüfsumme nicht berechnet wurde. Dies sollte aber nur mit bedacht eingesetzt werden, da IP keine Prüfsumme auf die IP-Daten unterstützt und somit die UDP-Prüfsumme das einzige Mittel ist, die Übertragung der Daten zu sichern. Wird die Prüfsumme berechnet, so beinhaltet diese neben dem UDP-Header und Daten auch noch einen Pseudo-Header (siehe Bild 4-17).

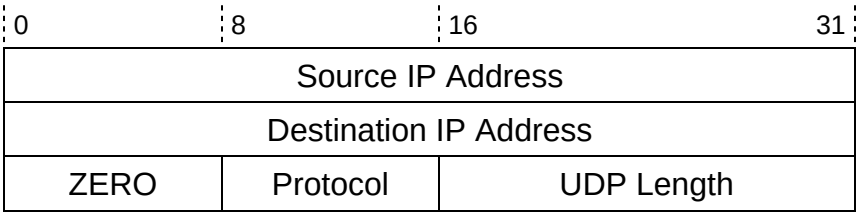



*Bild 4-17 Pseudoheader zur Berechnung der UDP-Prüfsumme*

Die Prüfsumme wird gebildet, indem zuerst die Prüfsumme im Datagramm auf 0 gesetzt und dann die 16-Bit Einer-Komplementsumme über Pseudo-Header, UDP-Header und UDP-Daten gebildet wird. Der Sinn des Pseudo-Headers ist die Kontrolle, ob das UDP-Datagramm auch das korrekte Ziel erreicht hat. Das Feld **Protocol** des Pseudo-Headers beinhaltet den Protokoll-Typ (17 für UDP) und das Feld **UDP-Length** beinhaltet die Länge des UDP-Datagramms ohne Pseudo-Header.

### 4.5.2.2 Verteilung der UDP-Nachrichten auf die Ports

Die UDP-Software muss ein Multiplexing bzw. Demultiplexing der UDP-Datagramme leisten. Zum einen muss die UDP-Software von verschiedenen Programmen UDP-Datagramme entgegennehmen und diese an die IP-Software weitergeben. Zum anderen müssen die UDP-Datagramme, die von der IP-Software an die UDP-Software gegeben werden, abhängig von der Portnummer an die entsprechenden Applikationen verteilt werden.

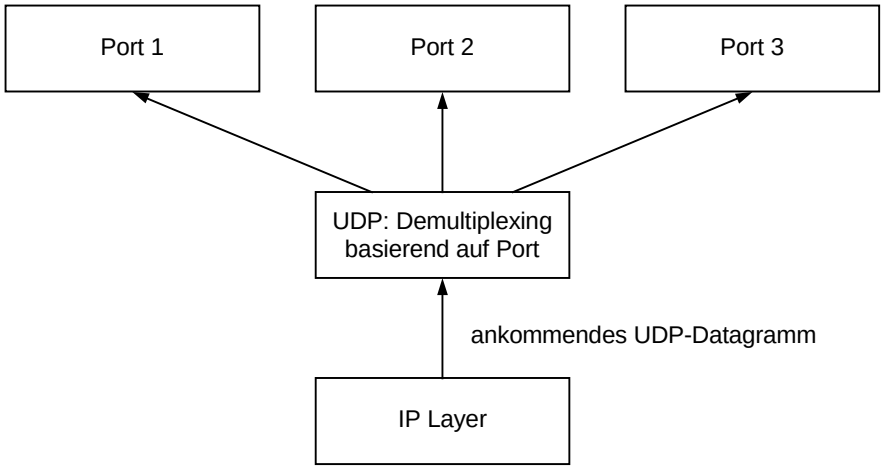



*Bild 4-18 Verteilung von Datagrammen durch UDP: Demultiplexing*

In den meisten Implementierungen stellt das Betriebssystem einer Applikation, die einen Protokoll-Port benutzt, eine Queue zur Verfügung. In diese Warteschlange schreibt die UDP-Software die Nachrichten für die Applikation, wenn ein UDP-Datagramm mit der entsprechenden Port-Nummer empfangen wird. Somit ist eine Zwischenspeicherung der Daten gewährleistet, wenn die Applikation einige Zeit lang keine Daten entgegen nehmen kann.

### 4.5.2.3 Reservierte UDP-Portnummern

Eine Applikation muss, um Daten über UDP an eine andere Applikation verschicken zu können, die Portnummer dieser Applikation kennen. Für UDP sind derzeit einige Portnummern weltweit eindeutig reserviert[3]. Des weiteren stehen eine große Anzahl von Portnummern für lokale Applikationen zur Verfügung. In Tabelle 4-1 sind einige dieser reservierten Portnummern aufgeführt. Für eigene Applikationen sollten Portnummern größer 1024 verwendet werden.

| Portnummer | Bezeichnung (UNIX) | Beschreibung |
|---|---|---|
| 53 | DNS | Domain Name Server |
| 69 | TFTP | Trivial File Transfer Protocol |
| 161 | Snmp-trap | SNMP net monitor |

*Tabelle 4-1 Beispiele für reservierte UDP-Portnummern*

[3] die sogenannten "well-known ports"

## 4.5.3 TCP – Transmission Control Protocol

Definiert in RFC 793 (September 1981).

Aus der Schicht 3 des TCP/IP Modells, der Transportschicht, wurde im vorherigen Kapitel UDP vorgestellt. UDP ist ein verbindungsloses Protokoll, welches keine gesicherte Übertragung von Daten über ein Netzwerk garantiert. Das zweite, sehr wichtige Protokoll aus der Schicht 3 des TCP/IP-Modells ist das Transmission Control Protocol (TCP). TCP ist ein verbindungsorientiertes Protokoll, welches eine gesicherte Datenübertragung garantiert und auf dem Stream-Mechanismus basiert.

### 4.5.3.1 Das Stream-Konzept

Verteilte Applikationen müssen oft sehr große Datenmengen zwischen den Rechnern austauschen. Eine solche Applikation auf der Basis eines ungesicherten, verbindungslosen Protokolls wie z.B. UDP würde bedeuten, dass die Programmierer in der Applikation Mechanismen zur Fehlererkennung und zur Wiederholung der Daten einbauen müssten. Da dies bei jeder weiteren Applikation wieder neu programmiert werden müsste, wurden diese Funktionalitäten von der Applikations- in die Transportschicht verlagert und ein gesichertes, stream-orientiertes Protokoll – das Transmission Control Protocol – entwickelt.

Beim Austausch großer Datenmengen zwischen zwei Programmen spricht man von einem Datenstrom von Bits bzw. Bytes (Octets). Der Stream-Service hat dabei die Aufgabe, dem Empfänger die Daten in genau derselben Reihenfolge zukommen zu lassen, wie diese vom Sender verschickt werden.

### 4.5.3.2 Weitere wichtige Eigenschaften des TCP-Protokolls

**Virtuelle Verbindung:**

Vor dem Beginn der Kommunikation informieren sowohl der Sender als auch der Empfänger das Betriebssystem, dass eine Verbindung aufgebaut werden soll. Einer der Partner sendet eine Anforderung, welche vom anderen Kommunikationspartner akzeptiert werden muss. Nach der Durchführung der Verbindungsaufnahme wird die Applikation des Kommunikationspartners informiert, dass der Datenaustausch beginnen kann. Im Fehlerfall, wenn die Verbindung z.B. unterbrochen wird, wird dies durch beide Kommunikationspartner erkannt und an die Applikation gemeldet. Die Verbindung ist aber nur virtuell, da das darunter liegende IP-Protokoll weiterhin auf dem Datagramm-Service beruht.

**Gepufferte Datenübertragung:**

Der Protokollsoftware steht es frei, den Datenstrom in Pakete zu unterteilen, unabhängig von den Datenpaketen, welche die Applikation übergibt. Wichtig ist nur, dass die Daten wieder in derselben Reihenfolge zusammengesetzt werden. Sendet eine Applikation z.B. ein Octet nach dem anderen, so kann, um eine höhere Effizienz zu erzielen und die Netzwerkbelastung zu minimieren, die Protokoll-Software solange Daten sammeln, bis ein genügend großes Datagramm entsteht. Entsprechend können auch extrem große Datenpakete in kleinere Blöcke unterteilt werden.

Für Applikationen, die einen Transfer der Daten auch in kleineren Einheiten benötigen, wurde der PUSH-Mechanismus eingeführt, der eine Datenübertragung unmittelbar anregt.

**Full-Duplex Verbindung:**

Verbindungen über TCP ermöglichen eine Full-Duplex Kommunikation, d.h. beide Applikationen können gleichzeitig senden. Eine Full-Duplex Verbindung besteht aus zwei unabhängigen Streams in gegensätzlicher Richtung. Beide Streams können unabhängig voneinander geschlossen werden, was die Verbindung zu einer Half-Duplex Verbindung macht.

### 4.5.3.3 Gewährleistung der gesicherten Übertragung

Um die Übertragung zu gewährleisten, wurde beim TCP-Protokoll der Mechanismus „acknowledgement with retransmission“ gewählt. Jedes Paket wird vom Empfänger bestätigt. Bleibt eine Bestätigung aus, schickt der Sender das Paket erneut. Hierzu muss der Sender das Paket solange aufbewahren, bis eine Bestätigung vom Empfänger eintrifft. Weiterhin muss der Sender einen Timer starten, um gegebenenfalls ein Packet erneut zu senden, falls innerhalb der eingestellten Zeit keine Bestätigung angekommen ist.

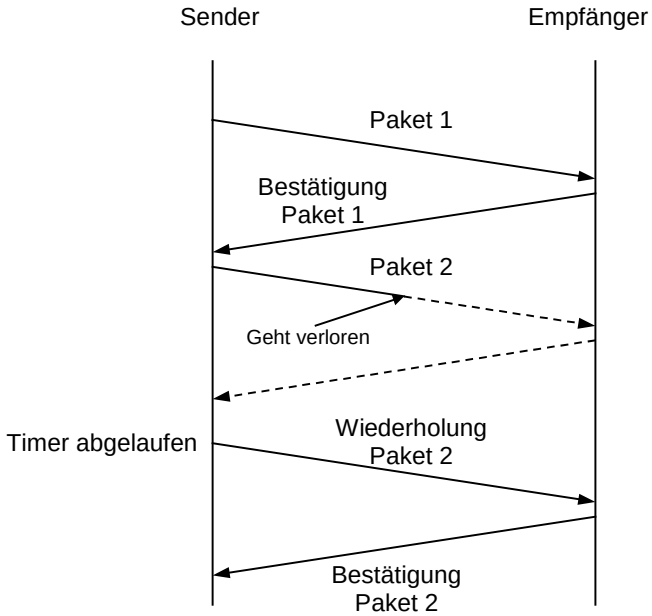



*Bild 4-19 Bestätigung und Wiederholung von Datenpaketen bei TCP*

Bei dieser Vorgehensweise kann es natürlich auch vorkommen, dass ein Packet beim Empfänger doppelt ankommt, wenn z.B. die Bestätigung vom Empfänger verloren geht. Deshalb müssen Sequenznummern eingeführt werden, damit der Empfänger gegebenenfalls doppelt empfangene Pakete eliminieren kann. Die Sequenznummern dienen ebenfalls zur korrekten Aneinanderreihung der Daten im Empfänger.

### 4.5.3.4 Die „Sliding-Window“ Technik

Der im Kapitel zuvor beschriebene Mechanismus „acknowledgement with retransmission“ erhöht zwar die Übertragungssicherheit enorm, wirkt sich aber in dieser einfachen Form sehr negativ auf die Bandbreitenausnutzung des Netzes aus. Der Sender muss nach dem Senden immer warten, bis der Empfänger eine Bestätigung des Paketes geschickt hat. Somit wird von der Full Duplex Verbindung zu einer Zeit immer nur 1 Kanal benutzt.

Mit der Sliding-Window Technik wird die Bandbreite effektiver ausgenutzt. Dem Sender wird dabei ermöglicht, mehrere Pakete zu verschicken, bevor er eine Bestätigung erwartet.

Stellt man sich die zu transportierenden Daten als eine Sequenz von Paketen vor, so wird zu Beginn der Übertragung ein Fenster einer definierten Länge darüber gelegt (siehe Bild 4-20).

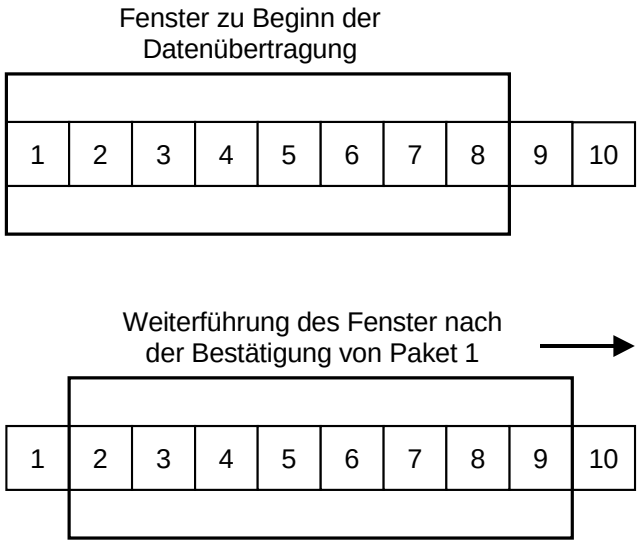



*Bild 4-20 Das Sliding-Window Prinzip mit einer Größe von 8 Paketen*

Dem Sender ist nur erlaubt, die Anzahl der Pakete zu versenden, die sich innerhalb des Fensters befinden, ohne auf eine Bestätigung zu warten. Das Fenster wird erst dann weiter bewegt, wenn das jeweils erste Paket bestätigt wird.

Das Protokoll muss sich hierzu jedes nicht bestätigte Paket merken und einen Timer für jedes abgeschickte Paket betreiben, um Pakete, die nicht in der vorgegebenen Zeit bestätigt wurden, erneut zu verschicken.

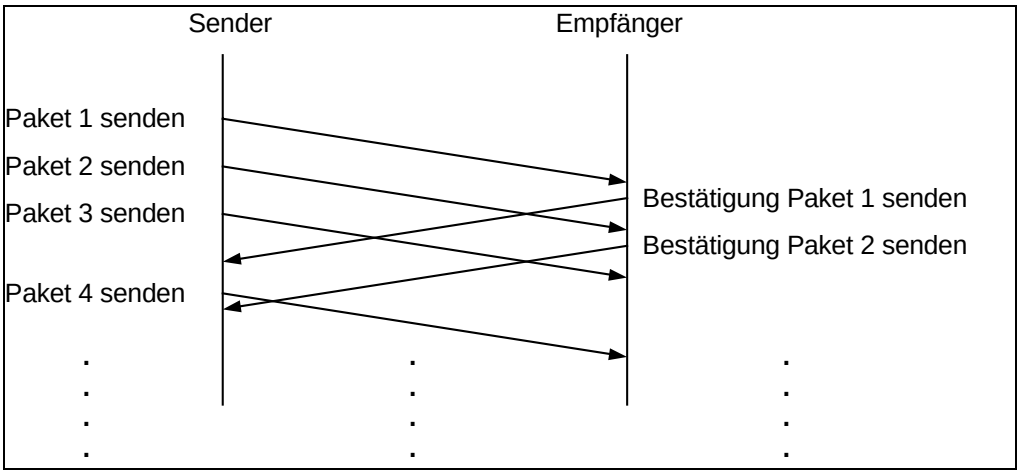



*Bild 4-21 Beispiel für das Senden von Daten bei einer Window-Size von 3*

Bei TCP wird nicht mit der Einheit Pakete gerechnet. Vielmehr wird dort direkt auf dem Datenstrom von Octets gearbeitet und ebenso die Fenstergröße in Anzahl Octets angegeben. Dies ist sinnvoll, da die Sequenznummern bei der Datenübertragung ebenfalls in der Einheit der Octets hochgezählt werden, und somit die Größe der Segmente flexibel bleibt.

Die TCP-Software im Sender hat drei Zeiger auf den Datenstrom zu verwalten. Der erste Zeiger markiert die linke Seite des Fensters, das die bestätigten von den unbestätigten Octets trennt. Ein zweiter Zeiger markiert die rechte Seite des Fensters und trennt damit die Octets, die ohne Bestätigung noch gesendet werden dürfen von den Octets, die erst bei einer Weiterführung des Fensters an der Reihe sind. Der dritte Zeiger liegt innerhalb des Sendefensters und markiert die Grenze zwischen den gesendeten und noch nicht gesendeten Octets.

#### 4.5.3.5 Ports und Verbindungen

Wie UDP benutzt TCP den Mechanismus der Ports, um einzelne Prozesse voneinander zu unterscheiden. Die Ports bei TCP sind allerdings um einiges komplexer, da es im Gegensatz zu UDP virtuelle Verbindungen gibt. Eine Verbindung ist dabei definiert durch ein Paar von Endpunkten.

Ein Endpunkt einer Verbindung setzt sich aus der IP-Adresse und der Portnummer zusammen, z.B. (128.10.2.30, 2000).

Eine Verbindung setzt sich dann aus 2 solcher Endpunkte zusammen, beispielsweise durch (128.10.2.30, 2000) und (53.1.182.99, 1500).

Zu einem Endpunkt können zu einer Zeit mehrere Verbindungen existieren. Dies ist möglich, da TCP eingehende Nachrichten mit der Verbindung statt nur mit der Portnummer in Zusammenhang stellt. Ein Programmierer benötigt also für ein Programm, welches mit mehreren anderen Applikationen kommunizieren will, lediglich eine Portnummer und nicht für jede Verbindung eine andere.

### 4.5.3.6 Variable Window-Size

Zusätzlich zu der eingeführten Sliding-Window Technik ist es bei TCP auch noch möglich, die Window-Size während der Laufzeit zu ändern. Jede Bestätigung vom Empfänger enthält eine Angabe darüber, wie viele Octets der Empfänger bei weiteren Sendungen zu Empfangen bereit ist (Window-Size). Dies hat den Vorteil, dass ein Empfänger so z.B. einem Empfangspufferüberlauf rechtzeitig vorbeugen kann. Sendet der Sender schneller, als der Empfänger die Daten verarbeiten kann, so verringert der Empfänger die Window-Size so weit, dass der Puffer nicht überläuft. Im Extremfall kann die Window-Size sogar auf 0 gesetzt werden, was einer Aufforderung zum Stoppen der Datenübertragung gleichkommt.

Dieser Mechanismus zur Flusskontrolle ist entscheidend, da in einem IP-Netz Maschinen unterschiedlicher Größe und Geschwindigkeit vorkommen können.

### 4.5.3.7 Das Format eines TCP-Paketes

Die Einheit eines TCP-Paketes wird meist als Segment bezeichnet. In Bild 4-22 ist das Format eines solchen Segments aufgeführt.

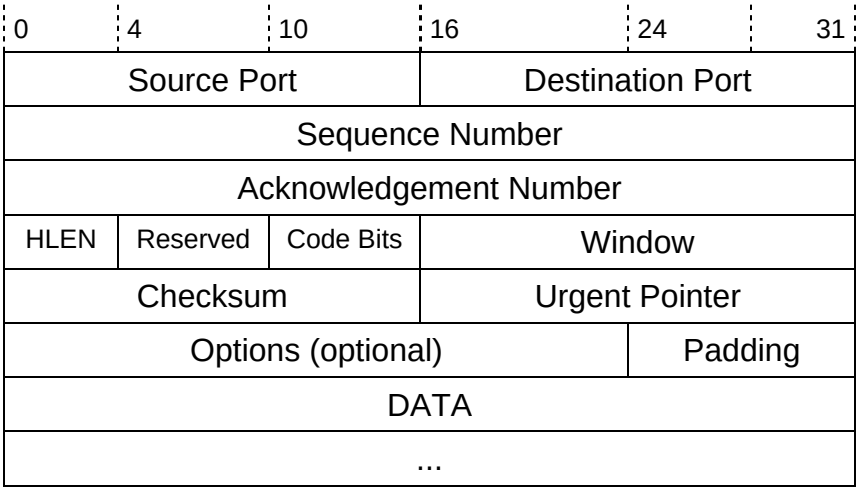



*Bild 4-22 Format eines TCP-Segments*

Der TCP-Header enthält die Identifikations- und Steuerinformationen eines Segments.

**Source- und Destination Ports:**

Enthalten die Portnummern der Applikationen

**Sequence Number:**

Enthält die Position des Pakets im Octet-Stream.

**Acknowledgement Number:**

Enthält die Sequenznummer des Octets, welches der Empfänger als nächstes empfangen will.

**HLEN:**

Die Länge des Headers in 32-Bit Werten (da Options variabel ist).

**Code Bits:**

| Bits (von links nach rechts) | Wenn gesetzt (=1) |
|---|---|
| URG | Urgent-Pointer ist gültig |
| ACK | Bestätigung (Acknowledge) eines Segments |
| PSH | Dies ist ein Push-Segment |
| RST | Reset der Verbindung |
| SYN | Synchronisation der Sequenz-Nummern |
| FIN | Ende der Übertragung |

**Window:**

Angabe der Größe des Empfangsfensters (Einheit in Octets und vielfaches der Segment-Größe)

**Checksum:**

Die Berechnung der Prüfsumme erfolgt, indem zuerst (wie bei UDP) ein Pseudo-Header vor das TCP-Segment gestellt wird (siehe Bild 4-23).

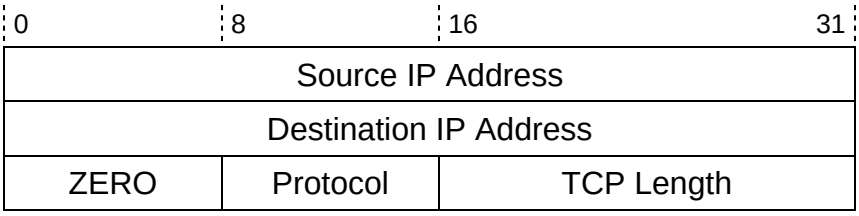



*Bild 4-23 Pseudo-Header für die Berechnung der TCP-Prüfsumme*

Der Pseudo-Header hat den Sinn festzustellen, ob das TCP-Paket den korrekten Empfänger erreicht hat. Das Feld **Protocol** enthält für TCP den Wert 6. Das Feld **TCP-Length** ist die Länge des gesamten TCP-Segments inklusive des Headers, aber ohne Pseudo-Header. Berechnet wird die Prüfsumme als 16-Bit Einerkomplement-Summe.

**Urgent-Pointer:**

In manchen Fällen ist es notwendig, Out-of-Band Daten zu verschicken, die, selbst wenn im Empfangspuffer des Empfänger noch Daten vorliegen, sofort an die Applikation weitergegeben werden. Dies ist z.B. bei einem Remote Login notwendig, wenn ein Abbruch des Login-Vorgangs erzwungen werden soll. Wenn ein Urgent-Pointer von TCP empfangen wird, muss TCP der Empfänger-Applikation den Urgent-Modus mitteilen, genauso, wie eine Rückkehr in den Normal-Modus nach Abschluss des Empfangs der Urgent-Pakete mitgeteilt werden muss. Wenn das URG-Bit gesetzt ist, weist der Urgent-Pointer auf die Position in den Daten innerhalb des Segments, wo die Urgent-Daten enden.

**Options:**

Mit dem Options-Feld wird die maximale Größe des TCP-Segments zwischen Sender und Empfänger abgestimmt. Dies ist zur Ausnutzung der maximalen Bandbreite eines Netzes wichtig.

### 4.5.3.8 Bestätigung von Daten

Der Empfänger ordnet empfangene Daten anhand ihrer Sequenznummer, um eine exakte Kopie des gesendeten Datenstroms herzustellen. Zu jeder Zeit hat der Empfänger 0 oder mehr Octets (vom Streamanfang zählend) kontinuierlich korrekte Daten empfangen. Es können aber auch Teile von Daten korrekt vorhanden sein, die außer der Reihe empfangen wurden. Der Empfänger bestätigt aber immer nur das letzte zusammenhängende, korrekt empfangene Paket. Dies wird dadurch gekennzeichnet, dass der Empfänger mit der Acknowledge-Number die Sequenznummer des nächsten zu empfangenen Octets angibt. Dies soll anhand eines Beispiels erläutert werden. Ein Sender beginnt bei der Sequenznummer 100 mit einer Window-Size von 500 Octets zu senden.

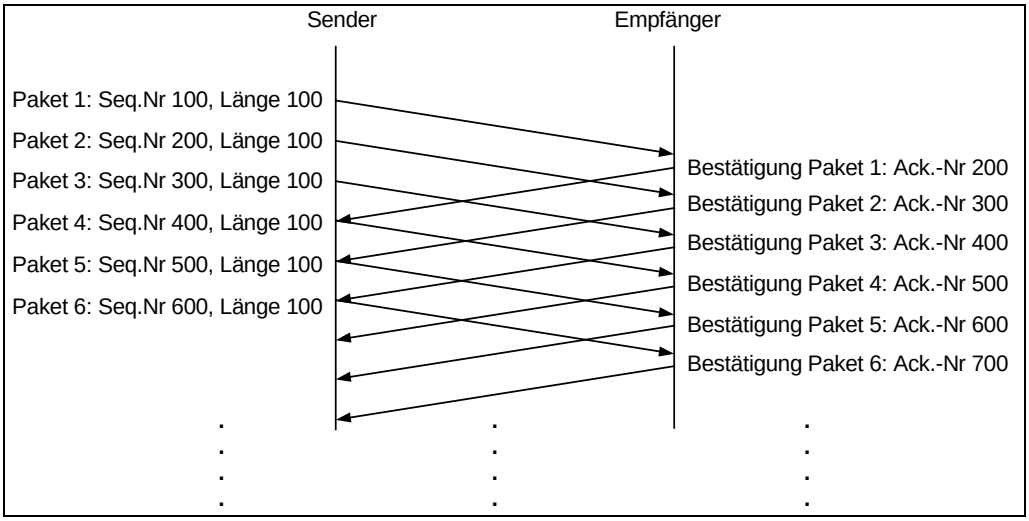



*Bild 4-24 Beispiel für die Sequenz- und Acknowledge-Nummern beim Datenaustausch*

Geht z.B. das Paket Nr. 2 verloren, so sendet der Empfänger weiterhin Bestätigungen mit der Acknowledge-Nummer 200, da dies die nächste folgende Sequenznummer des Octets ist, die auf den bisher vorliegenden korrekt empfangenen Datenbereich folgt. Der Empfänger sendet die ACK-Nr. 200 auch, wenn weitere Pakete korrekt empfangen werden. Wenn im Sender das Timeout für die Bestätigung des 2. Paketes und der folgenden erfolgt, muss dieser entscheiden, ob er alle 5 Pakete oder nur das erste Paket des aktuellen Fensters verschickt. Alle 5 zu wiederholen ist möglich, wäre natürlich nicht sehr effektiv. In diesem Fall würde der Empfänger alle Pakete bestätigen, die Pakete 3 – 6 aber verwerfen (für den Fall, dass Paket Nr. 3 – 6 vorher korrekt empfangen wurden). Die ACK-Nr. für alle neu gesendeten Pakete wäre immer 700. Wiederholt der Sender nur Paket Nr. 2 und hat der Empfänger die Pakete Nr. 3 – 6 korrekt empfangen, antwortet der Empfänger mit der ACK-Nr. 700 und das Sendefenster kann im Sender um 500 Octets verschoben werden.

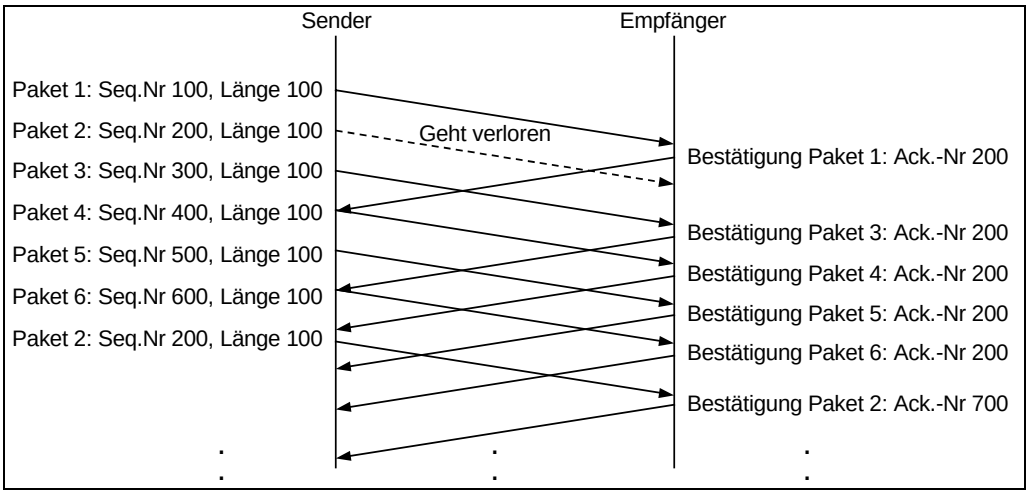



*Bild 4-25 Beispiel für die Abfolge der Acknowledge-Nummern beim Verlust eines Paketes*

Das Senden von nur einem Paket macht das Protokoll aber wieder zu einem einfachen „acknowledgement protocol“ und die Vorteile der Fenstersteuerung sind verloren, da wieder nur 1 Paket versendet wird und auf die Bestätigung von nur diesem einen Paket gewartet wird.

### 4.5.3.9 Aufbau einer TCP-Verbindung

Für den Aufbau einer TCP-Verbindung wird ein 3-Wege Handshake-Verfahren benutzt.

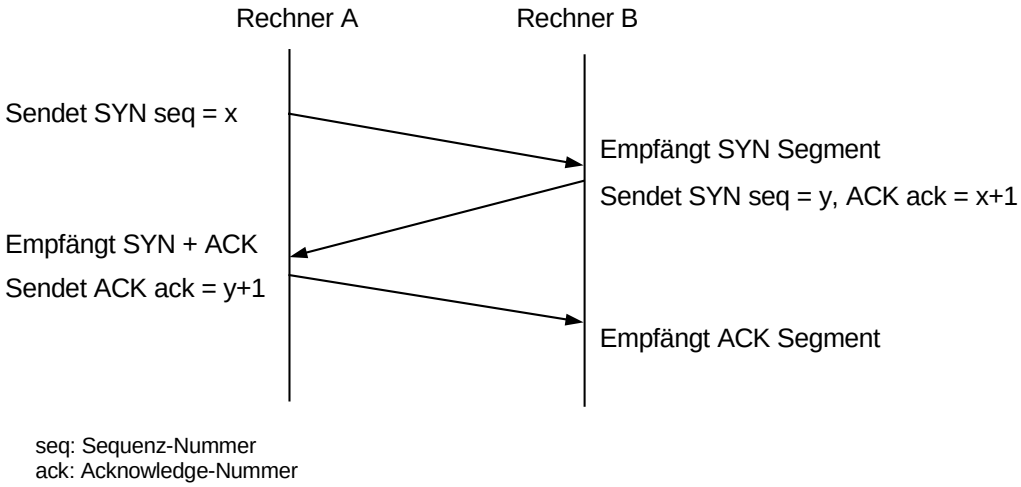



*Bild 4-26 Aufbau einer TCP-Verbindung mit dem 3-Wege Handshake-Verfahren*

Rechner A sendet ein Segment mit gesetztem SYN-Bit. Rechner B antwortet darauf mit einem Segment mit gesetztem SYN- und ACK-Bit. Hierbei ist zu beachten, dass die Ack.-Nr. x+1 die Bestätigung auf die Nachricht mit der Seq.-Nr. x ist. Die abschließende Nachricht ist die Bestätigung (gesetztes ACK-Bit) des Rechners A.

In den meisten Fällen wartet die TCP-Software passiv auf die Verbindungsaufnahme von einem anderen Rechner (in Bild 4-26 wartet der Rechner B). Jedoch ist es beim 3-Wege Handshake-Verfahren auch möglich, dass beide zeitgleich eine Verbindungsaufnahme durchführen.

Beim Verbindungsaufbau werden zwei, für den Datenaustausch wichtige Voraussetzungen erfüllt

- es wird garantiert, dass beide Seiten zum Datenaustausch bereit sind und
- beide Seiten haben die Startwerte der Sequenznummern ausgetauscht.

Wichtig ist der Austausch der Startwerte vor allem dann, wenn schon das erste Datenpaket, welches geschickt wurde, verloren geht. Auf alle folgenden Datenpakete muss der Empfänger mit einer Bestätigung und der entsprechenden Ack-Nr. antworten. Dies ist immer die Ack.-Nr., die der Seq.-Nr. des ersten Octets des ersten, verlorengegangenen Pakets entspricht. Für den Fall nach der Verbindungsaufnahme in Bild 4-26 ist die Ack.-Nr bei Verlust des ersten Packets immer x+1 (falls A Sender und B Empfänger ist).

#### 4.5.3.10 Abbau einer Verbindung

Zum Abbau einer Verbindung wird ein modifiziertes 3-Wege Handshake-Verfahren benutzt. Wie bereits erwähnt, ist eine TCP-Verbindung eine Full-Duplex Verbindung mit zwei unabhängigen, in gegensätzlichen Richtung fließenden Datenströmen. Diese Richtungen können unabhängig voneinander geschlossen werden. Wenn eine Applikation der TCP-Software meldet, dass keine Daten mehr zum Senden vorhanden sind, so wird die Senderichtung beendet.

TCP beendet dabei das Senden der Daten, wartet bis das letzte Segment vom Empfänger bestätigt wurde und sendet dann ein Segment mit gesetztem FIN-Bit. Dieses Segment wird vom Empfänger bestätigt (gesetztes ACK-Bit) und die Empfangsapplikation wird darüber informiert, dass keine weiteren Daten mehr kommen werden.

Ist eine Verbindungsrichtung einmal beendet, so verweigert die TCP-Software die Annahme weiterer Daten, die darüber transportiert werden sollen.

Die Verbindung in Gegenrichtung wird auf dieselbe Art beendet. Erst wenn beide Verbindungsrichtungen beendet sind, verwirft die TCP-Software die Information der Verbindung und beendet die Kommunikation mit der Applikation.

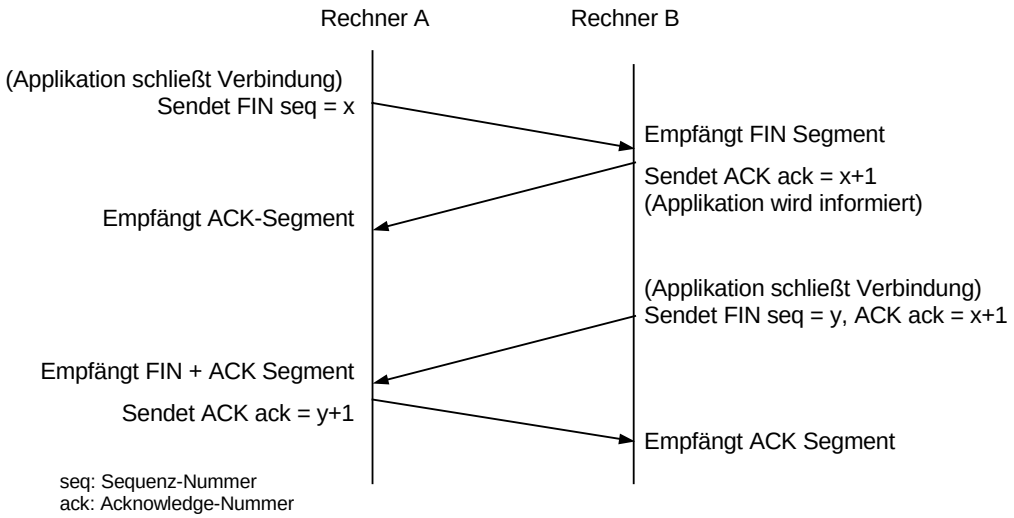



*Bild 4-27 Abbau einer TCP-Verbindung*

Der Unterschied zum 3-Wege Handshake-Verfahren ist, dass der Empfänger des erste FIN-Segmentes nicht ebenfalls sofort ein FIN-Segment zurückschickt. Vielmehr wird das erste FIN-Segment lediglich bestätigt und durch das Informieren der Applikation die weiteren Schritte der Applikation überlassen.

#### 4.5.3.11 Reservierte Portnummern

Wie auch bei UDP gibt es bei TCP fest vereinbarte Portnummern. Bei TCP sind die ersten 256 Portnummern reserviert. In Tabelle 4-2 ist eine Auswahl der reservierten Ports zu sehen. Für eine Applikation wird empfohlen, Portnummern über 1024 zu verwenden.

| Portnummer | Bezeichnung | Beschreibung |
|---|---|---|
| 21 | FTP | File Transfer Protocol |
| 23 | Telnet | Terminal Connection |
| 25 | SMTP | Simple Mail Transport Protocol |
| 53 | DNS | Domain Name Server |
| 80 | HTTP | Hypertext Transfer Protocol |
| 103 | X.400 | X.400 Mail Service over IP |
| 113 | Auth | Authentication Service |

*Tabelle 4-2 Reservierte Portnummern bei TCP*

# 4.6 Anwendungsschicht

## 4.6.1 DNS – Domain Name System

Definiert in RFC 1034 (Nov. 1987) – concept and facilities

RFC 1035 (Nov. 1987) – implementation and specification

In den bisher besprochenen Protokollen wurde zur Adressierung eines Rechners eine 32-Bit Internet-Adresse verwendet. Diese Adressen sind zwar eindeutig vergeben, es ist aber nicht jedermanns Sache, diese Zahlenfolgen auswendig zu lernen. Viel einprägsamer ist ein Name, den man an einen Rechner vergibt.

In diesem Kapitel wird ein Namensschema vorgestellt, nach dem Rechner benannt werden und ein Mechanismus gezeigt, mit dem diese Namen auf IP-Adressen abgebildet werden.

### 4.6.1.1 Maschinennamen

Die Entstehungsgeschichte der Rechnerkommunikation ging von einfachen Punkt-zu-Punkt Verbindungen, bei denen Hardwareadressen verwendet wurden, zu kleinen lokalen Netzen, bei denen Rechner nach ihrem Einsatzgebiet, z.B. Entwicklung, Accounting, ... benannt wurden.

Hierzu musste auf jedem Rechner, der an der Kommunikation teilnehmen wollte, eine Zuordnungstabelle gepflegt werden, in der ein Rechnername einer Rechneradresse zugeordnet wird. Eine solche Hosttabelle ist z.B. in UNIX in der Datei `/etc/hosts` hinterlegt.

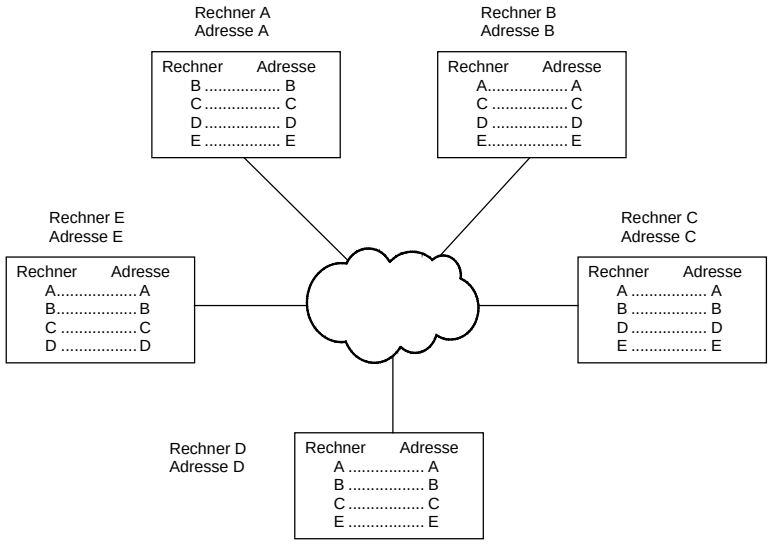



*Bild 4-28 Zuordnungstabellen von Namen zu Adressen auf jedem Rechner*

Je mehr Maschinen aber an dieser Kommunikation teilnahmen, umso größer war der Aufwand diese Tabellen zu pflegen. Ein weiterer, aber nicht unwichtiger Punkt ist die Auswahl des Namens. Der Namensraum ist begrenzt und so kann es vorkommen, dass Namen doppelt vergeben werden.

#### 4.6.1.2 Hierarchische Namen

In den Anfängen des Internet wurden Namen in der in Bild 4-28 gezeigten flachen Struktur vergeben. Die Vergabe erfolgte dabei durch eine zentrale Stelle, dem Network Information Center (NIC) (so wird verhindert, dass Namen doppelt vergeben wurden). Später wurde das NIC durch das INTERnet Network Information Center (INTERNIC) ersetzt.

Ziemlich bald wurde erkannt, dass der Namensraum durch den rapiden Zuwachs der an das Internet angeschlossenen Rechner erschöpft worden wäre[4]. Somit wurde der hierarchisch aufgeteilte Namensraum beschlossen.

Die Hierarchie der Namen im Internet ist gegliedert in Domains (Gebiete). Ein Name setzt sich aus einer Sequenz von Teilnamen, getrennt durch Punkte, zusammen. Z.B. enthält der Name

`rz.fht-esslingen.de`

3 Teile: `rz`, `fht-esslingen` und `de`. Jeder Teil entspricht daher einer Domäne. Die niedrigste Domäne ist `rz.fht-esslingen.de`. Die nächst höhere `fht-esslingen.de` (der Domänennamen der FH Esslingen) und die höchste `de` (der Domänennamen für Deutschland).

Der Rechnername selbst wird dann, getrennt durch einen Punkt, vor den Namen der Domäne, in der er sich befindet, geschrieben. So ist z.B. der Rechner `itix01.it.fht-esslingen.de` ein Rechner in der Domäne `it.fht-esslingen`.

#### 4.6.1.3 Namensvergabe im Internet

Im Internet wurde beschlossen, die folgenden Domänen der obersten Ebene einzuführen:

| | |
|---|---|
| COM | COMmercial |
| EDU | EDUcational |
| GOV | GOVernment |
| MIL | MILitary |
| DE | DEutschland |
| UK | United Kingdom |
| US | United States |
| ... | |

Da das Internet in den Anfängen nur in der USA eingesetzt wurde, ist das Namensschema dadurch sehr geprägt. So wurde anfänglich die oberste Domäne nur nach US-amerikanischen Gesichtspunkten eingeteilt (COM, EDU, GOV, MIL, ...). Mit

[4] Natürlich sind durch die Zeichen des Alphabets im Prinzip fast beliebig viele Kombinationen möglich, nur ist es gewollt, sinnvolle Namen und nicht kryptische Buchstabenkombinationen als Rechnernamen zu vergeben.

der weiteren Verbreitung wurden dann andere Länder über ihren ISO-Landesbezeichner mit aufgenommen (DE, UK, ...).

Die Verwaltung der Domänen wird von verschiedenen Organisationen vorgenommen. Die Domäne `de` wird z.B. durch die DENIC verwaltet, die Domänen `com`, `org` oder `net` verwaltet die INTERNIC. Es steht dabei jedem frei, sich bei einer solchen Organisation (gegen einen kleinen Unkostenbeitrag) einen Namen eintragen zu lassen (Informationen hierzu z.B. über die DENIC bei www.nic.de oder die INTERNIC bei rs.internic.net).

Es ist auch durchaus üblich, dass sich z.B. eine Firma mehrere Domains reserviert. So hat z.B. die Daimler Benz AG u.a. die Namen `Daimler-Benz.com`, `daimler-benz.de`, `daimler.de`, `dbag.com`, `dbag.de` reserviert.

Die Vergabe von Subdomains kann von dem jeweiligen Eigner der Domäne selber vorgenommen werden. So hat z.B. die FH-Esslingen u.a. die Subdomains `rz.fht-esslingen.de`, `it.fht-esslingen.de`, `wi.fht-esslingen.de` oder `mb.fht-esslingen.de` eingeführt.

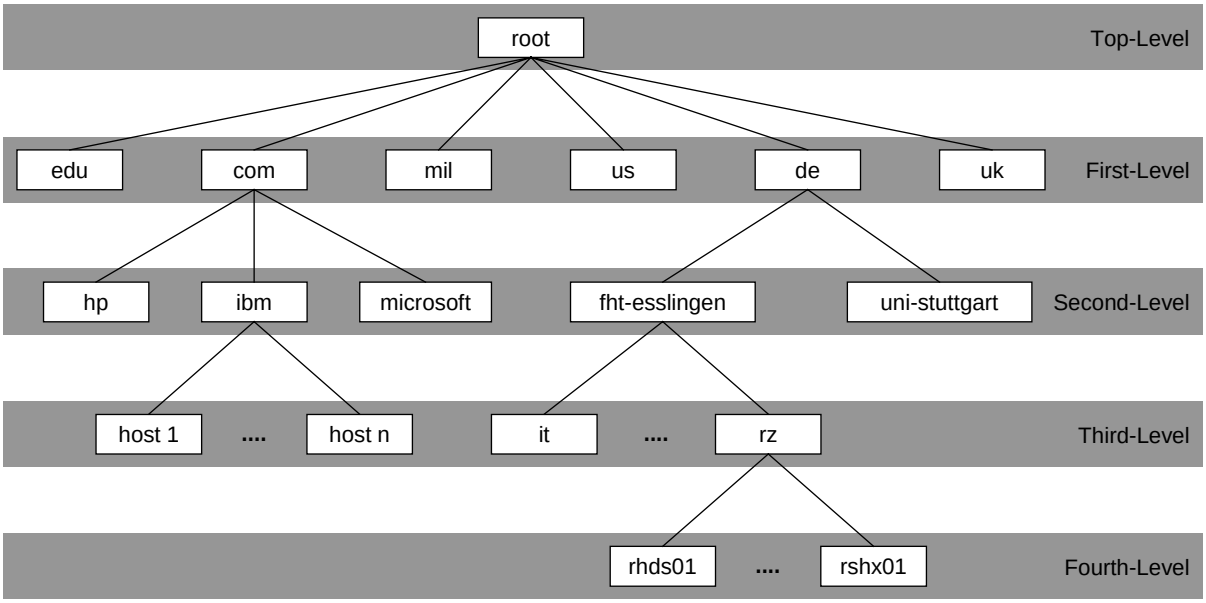



*Bild 4-29 Hierarchische Struktur der Domain Names*

In Bild 4-29 ist zu sehen, wie die Domänen weltweit aufgeteilt sind. Die oberste Domäne wird `root` oder auch nur `.` (Dot) genannt.

#### 4.6.1.4 Abbildung der Domain-Namen auf Adressen

In Bild 4-28 wurde die Pflege der Namen auf jedem Rechner gezeigt. Mit der Zunahme der Rechner im Netz wird diese Pflege sehr aufwendig und fehleranfällig. Jede neue Adresse muss auf jedem Rechner hinterlegt werden.

Um dieses Problem zu umgehen, wurden sogenannte Name-Server eingeführt. Ein Name-Server ist ein Programm, welches eine Übersetzung eines Namens in eine IP-Adresse vornehmen kann. Ein Rechnername muss bei der Verwendung eines Name-Servers nur 1 mal, und zwar im Name-Server, gepflegt werden. Rechner, die eine Adresse benötigen, fragen diese beim Name-Server ab. Ein Rechner muss also nicht mehr alle Rechneradressen kennen, sondern nur noch die des Name-Servers.

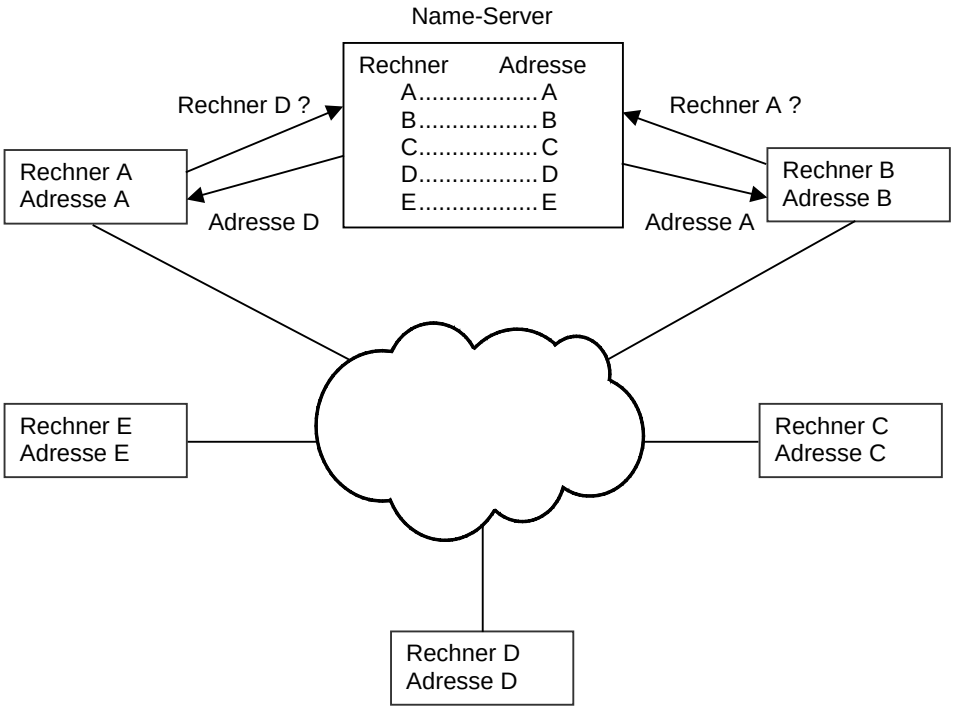



*Bild 4-30 Name-Server in einem Netz*

Dabei verwaltet ein Name-Server die Rechnernamen einer Domäne oder Subdomäne (es ist auch möglich, dass ein Name-Server mehrere Domänen und Subdomänen verwaltet). Für eine weltweite Kommunikation muss ein Netz von Name-Servern aufgebaut sein. Dieses Netz besteht aus mehreren Root-Servern (zur Zeit 9 Stück). An diesem Verbund von Root-Servern sind die Domänen-Server der First-Level-Domain (`de`, `com`, `edu`, ...) in der nächsten Ebene aufgehängt.

Eine weitere Ebene tiefer folgen dann die Name-Server der Second-Level-Domäne wie z.B. `fht-esslingen.de` oder `daimler-benz.de`.

An der FH-Esslingen gibt es, wie bereits erwähnt, die Domäne `fht-esslingen.de` und mehrere Subdomänen, z.B. `rz.fht-esslingen.de`, `it.fht-esslingen.de` oder `mb.fht-esslingen.de`. Diese werden alle gemeinsam von einem Name-Server (`rhhx01.fht-esslingen.de`) verwaltet.

#### 4.6.1.5 Kommunikation zwischen Clients und Name-Servern bzw. zwischen Name-Servern

Wenn ein Name-Server eine Abfrage von einem Rechner seiner Domäne erhält, prüft er zuerst, ob der zu übersetzende Name in seiner Domäne liegt (wichtig ist hier, dass ein Rechner, der einen Namen in eine Adresse übersetzen lassen will, immer den Name-Server abfragt, dessen Adresse (nicht sein Name!) in der TCP/IP-Software hinterlegt ist; in der Regel ist dies der lokale Name-Server der Domäne). Kann der Name-Server den Namen auflösen, schickt er die passende IP-Adresse an den fragenden Rechner zurück.

Ist dies nicht der Fall, so muss zwischen einer rekursiven und einer nicht-rekursiven Abfrage des Clients unterschieden werden. Mit einer rekursiven Abfrage verlangt ein Client von einem Name-Server eine vollständige Auflösung des Namens. Der Name-Server hat also die Aufgabe, solange nach einem Name-Server zu suchen, bis die Abfrage vollständig beantwortet ist. Eine nicht rekursive, iterative Abfrage bedeutet, dass der Client für die Abfrage weiterer Name-Server selbst zuständig ist, wenn der Name-Server die Abfrage des Clients nicht beantworten kann. In der Regel werden aber rekursive Abfragen durch den Client ausgeführt.

Der Algorithmus für die Abfrage eines Namens, den ein Name-Server nicht selbst auflösen kann, ist, dass sich der Name-Server sofort an einen der Root-Server wendet und dieser die Adresse des nächsten Name-Servers zurückgibt, der bei dieser Abfrage weiterhelfen kann. Dies geschieht so lange, bis der Name-Server gefunden ist, in dessen Domäne der Name enthalten ist, und dieser die passende IP-Adresse zurückgeben kann.

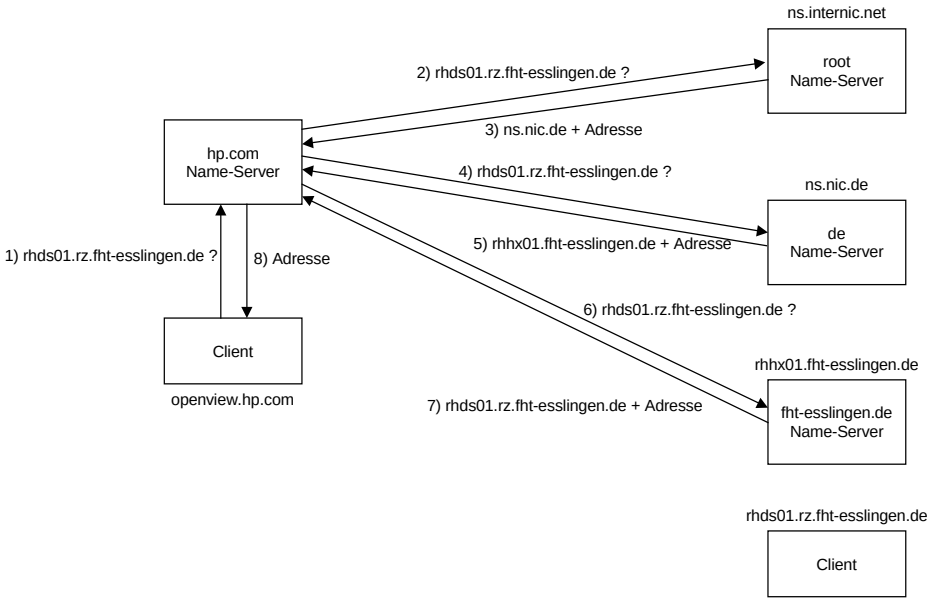



*Bild 4-31 Ablauf einer Abfrage eines nicht lokalen Rechnernamens*

Somit sind die einzigen Informationen, die ein Name-Server braucht, die IP-Adressen von einigen Root-Servern. Die Root-Server kennen nur die IP-Adressen der Domänen-Server, usw.

#### 4.6.1.6 Aliasnamen

Es ist auch möglich, für Rechner Aliasnamen zu vergeben. Mit einem Alias wird ein Rechnername mit einem weiteren, z.B. applikationsbezogenen Namen versehen. Dies kann beispielsweise für folgenden Anwendungsfall wichtig werden:

Als Beispiel soll ein verteiltes System aus 2 Server-Rechnern Server1 und Server2 existieren. Auf diesen Servern sind verschiedene Server-Prozesse eines Buchungssystems für eine Reisebürokette aktiv:

| **Server** | **Applikation** |
|---|---|
| Server1 | Flugauskunft, Flugbuchung |
| Server2 | Hotelreservierung |

An diese Server sind mehrere Tausend Clients angeschlossen. Damit diese Clients arbeiten können, muss zum Aufruf der jeweiligen Aktion ein Rechnername hinterlegt werden, d.h. Server1 für die Flugauskunft und Flugbuchung und Server2 für die Hotelreservierung. Ein Problem tritt aber dann auf, wenn Server1 im Laufe der Zeit überlastet ist und beschlossen wird eine der 2 Applikationen auf einen eigenen Server auszulagern. Dann muss in den tausenden von Clients dieser Rechnername geändert werden. Benutzt man aber Aliasnamen wie z.B.

| **Adresse** | **Name** | **Alias** |
|---|---|---|
| 134.108.56.60 | Server1 | Flugauskunft, Flugbuchung |
| 134.108.56.61 | Server2 | Hotelreservierung |

so ist es jederzeit möglich, einen neuen Rechner einzufügen

| **Adresse** | **Name** | **Alias** |
|---|---|---|
| 134.108.56.60 | Server1 | Flugauskunft |
| 134.108.56.61 | Server2 | Hotelreservierung |
| 134.108.56.62 | Server3 | Flugbuchung |

und einen der Aliasnamen von Server1 auf Server3 zu übertragen. Kein Änderungsaufwand bei den Clients ist nötig, wenn in den Clientapplikationen von Anfang an nur die Aliasnamen verwendet werden.

#### 4.6.1.7 Caching

Die Kosten und auch die Netzlast für die Auflösung nicht-lokaler Namen können extrem hoch werden. Aus diesem Grund besitzen die meisten Implementationen eines Name-Servers einen Cache für häufig benutzte Namen. Zusätzlich zu den Namen mit entsprechender Adresse wird in diesem Cache auch noch die Adresse des Name-Servers, von dem dieser Rechnername aufgelöst wurde, abgespeichert.

Fragt ein Client den lokalen Name-Server ab, prüft dieser zuerst, ob der Name zu seiner eigenen Domäne gehört. Ist dies nicht der Fall, so wird der Cache durchsucht. Erst wenn der gesuchte Name auch dort nicht enthalten ist, beginnt der Name-Server die Abfrage beim Root-Server.

Dem Client wird mitgeteilt, wenn ein Name aus dem Cache geholt wurde. Genügt dem Client diese evtl. veraltete Information nicht, so kann dieser den lokalen Name-Server auffordern, bei dem ebenfalls noch bekannten Name-Server anzufragen, ob die Namensauflösung noch korrekt ist. Da aber die Zuordnung von Name und IP-Adresse sehr selten geändert wird, ist dies kaum notwendig und der Client wird im Regelfall die Abfrage aus dem Cache akzeptieren.

#### 4.6.1.8 Inverse Abfragen

Es ist neben der bisher beschriebenen Abfrage auch möglich, eine inverse Abfrage an einen Name-Server zu stellen, d.h. man gibt eine IP-Adresse vor und erhält einen Rechnernamen zurück. Diese Form der Abfrage wird auch Pointer-Query genannt.

Um einen Pointer-Query zu generieren, muss die bekannte IP-Adresse

`aaa.bbb.ccc.ddd`

in umgekehrter Reihenfolge angegeben und anschließend der String `in-addr.arpa` angehängt werden:

`ddd.ccc.bbb.aaa.in-addr.arpa`

Die umgekehrte Reihenfolge deshalb, da das most significant Octet im Domänen Namen als letztes und in der IP-Adresse als erstes kommt.

#### 4.6.1.9 Das Kommando nslookup

Mit dem Kommando `nslookup` kann eine Abfrage an den Name-Server aus einer Shell des Betriebssystem durchgeführt werden. Das Kommando `nslookup` hat folgende Optionen:

`nslookup [-opt] [host] [server]`

| | |
|---|---|
| `-opt` | Setzen von Optionen, z.B. `–querytype=PTR` (oder nur `–q=PTR`) für einen Pointer-Query |
| `host` | der gesuchte Host |
| `server` | Angabe eines anderen als den default Name-Server |

Es ist auch möglich, im interaktiven Modus mit `nslookup` zu arbeiten. Hier sei aber auf das jeweilige Online-Manual verwiesen.

Hier einige Beispiele für die Benutzung von `nslookup` (die Aufrufe wurden auf dem Rechner `rshx01` ausgeführt):

```
nslookup rhds01.rz.fht-esslingen.de
>Name Server: rshx01.rz.fht-esslingen.de
>Address:     134.108.51.1
>
>Name:    rhds01.rz.fht-esslingen.de
>Address: 134.108.56.60
```

`nslookup rhds01` ergibt selbes Ergebnis wie oben, da die Angabe des Rechnernamens grundsätzlich um die der lokalen Domäne ergänzt wird.

```
nslookup www.microsoft.de
>Name Server: rshx01.rz.fht-esslingen.de
>Address:     134.108.51.1
>
>Name:    nt2.microsoft.de
>Address: 194.120.227.190
>Aliases: www.microsoft.de
```

Und zum Abschluss noch eine inverse Abfrage mit der Option PTR

```
nslookup –q=PTR 101.57.108.134.in-addr.arpa
>Name Server: rshx01.rz.fht-esslingen.de
>Address:     134.108.51.1
>
>101.57.108.134.in-addr.arpa   name=itix01.it.fht-esslingen.de
```

## 4.6.2 TELNET

Definiert in RFC 854 (Mai 1983)

Die Funktion Interactive Terminal Login wird in der TCP/IP-Protokollfamilie durch das TELNET-Protokoll realisiert.

TELNET erlaubt dem Benutzer (kann auch ein Anwendungsprogramm sein), eine TCP-Verbindung zu einem Login-Server auf einem entfernten System herzustellen. Dabei werden Eingabedaten vom Terminal direkt zur entfernten Maschine gesandt und in Gegenrichtung Ausgaben vom entfernten System zurück an das Terminal geleitet, so als sei dieses ein lokales Terminal des entfernten Systems (Bild 4-32). Dazu ist es erforderlich, dass das Betriebssystem der Server-Maschine eine, als "virtuelles Terminal" bezeichnete Schnittstelle unterstützt. Das virtuelle Terminal gestattet, von einem Programm aus Zeichen einzuschleusen als ob sie von einem realen Terminal kämen, und umgekehrt für ein Terminal bestimmte Ausgaben zu übernehmen.

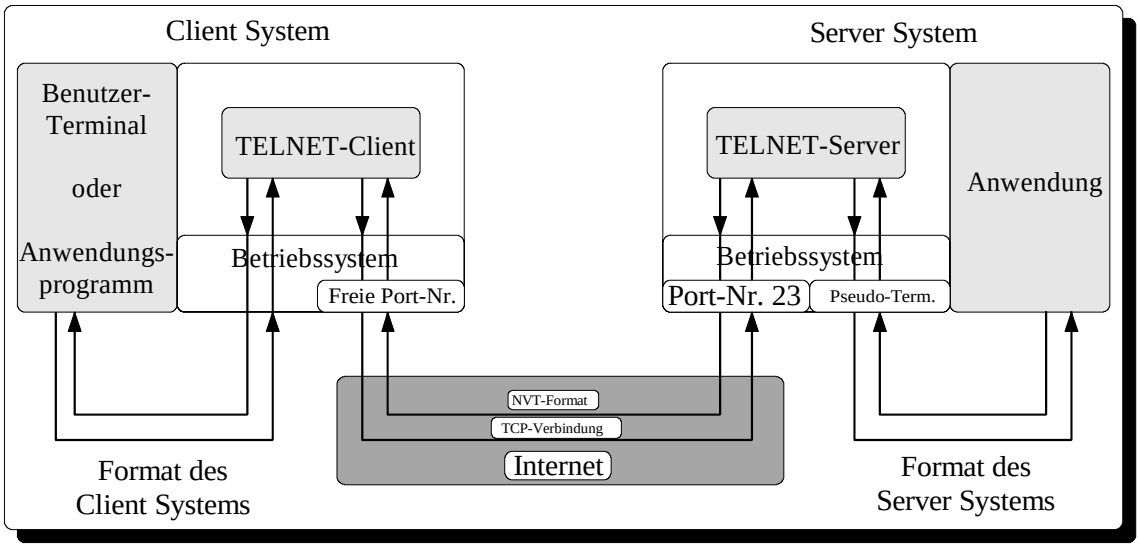



*Bild 4-32 TELNET Client-Server Beziehung*

TELNET hat zwei wichtige Komponenten:

1. Um die mögliche Heterogenität in den Terminals und in den beteiligten Systemen zu beherrschen, ist das Network Virtual Terminal (NVT) als Standard-Interface zwischen den beteiligten Systemen definiert. Das Standard-NVT-Format wird auf der Internet-Verbindung zwischen Client und Server benutzt. Im Client werden die vom Terminal kommenden und im lokalen System gültigen Zeichen vor dem Weitertransport zum Server auf das NVT-Format umgesetzt, und umgekehrt werden die vom Server im NVT-Format ankommenden Sequenzen in das im Client-System gültige Format gewandelt. Der Server realisiert die entsprechenden Umwandlungen.
2. Es ist ein Mechanismus vorhanden, mit dessen Hilfe Client und Server Optionen für die TELNET-Verbindung aushandeln können (z.B., ob 7- oder 8-Bit ASCII-Code verwendet werden soll), und ein Satz an Standardwerten. Dieser Mechanismus ist symmetrisch, d.h. Client und Server sind beim Aushandeln von Optionen gleichberechtigt.

### 4.6.3 FTP - File Transfer Protocol

Definiert in RFC 959 (Oktober 1985).

FTP ist das Standard File Transfer Protokoll der TCP/IP-Protokollfamilie und setzt auf TCP als zuverlässige Transportverbindung auf.

FTP leistet nicht nur den bittransparenten Transfer von Dateien zwischen den beteiligten Systemen, sondern:

- FTP kann von Programmen aus benutzt werden. Die meisten Implementationen haben aber zusätzlich eine interaktive Schnittstelle, über die ein Benutzer über den FTP-Server im entfernten System mit dem Dateisystem dieses Rechners korrespondieren kann. Er kann etwa eine Liste aller Files in einem Verzeichnis (Directory) anfordern, aber auch konkrete Maßnahmen durchführen (z.B. ein neues Subdirectory einrichten).
- FTP verlangt zwingend, dass sich ein Benutzer durch eine User-ID eindeutig identifiziert und durch das Passwort legitimiert. Ohne dieses verweigert der Server jeden Zugriff auf das dortige Dateisystem.
- FTP erlaubt nicht nur einen bittransparenten Transport ganzer Dateien. Der Benutzer kann Datenformate angeben, z.B. ob die Inhalte binäre Zahlenwerte darstellen oder alphanumerische Zeichen sind, und, wenn es Zeichen sind, ob sie ASCII- oder EBCDIC-verschlüsselt sind. Beim Transfer werden die gegebenenfalls erforderlichen Umsetzungen (z.B. ASCII ⇔ EBCDIC) automatisch vorgenommen. Es ist zu beachten, dass bei solchen Umsetzungen Informationen verloren gehen können, und die Umsetzungen dann nicht mehr umkehrbar eindeutig sind (wenn etwa zwei Systeme unterschiedliche Gleitkommadarstellungen haben, so dass bei einer Umwandlung Genauigkeit verloren geht, so kann diese bei einer anschließenden Rückkonversion nicht wieder hinzugefügt werden).

Wie bei anderen Internet-Diensten auch, können die Dienste eines FTP-Servers gleichzeitig von mehreren Clients in Anspruch genommen werden. Anders als bei

den anderen Diensten werden für eine FTP-Sitzung u.U. aber mehrere Transportverbindungen zwischen einem Client und dem Server etabliert. Zu Beginn einer FTP-Sitzung wird zunächst eine Steuerverbindung aufgebaut, über die z.B. eine Authentisierungs-Prozedur abgewickelt wird und über die Steuerinformationen (auch Kommandos) laufen. Falls es zum Transfer einer Datei kommt, wird dafür eine separate Verbindung aufgebaut (vgl. Bild 4-33). Beide Verbindungen setzen auf TCP auf; bei der Steuerverbindung handelt es sich um eine TELNET-Verbindung mit reduzierter Funktionalität (keine aushandelbaren Optionen).

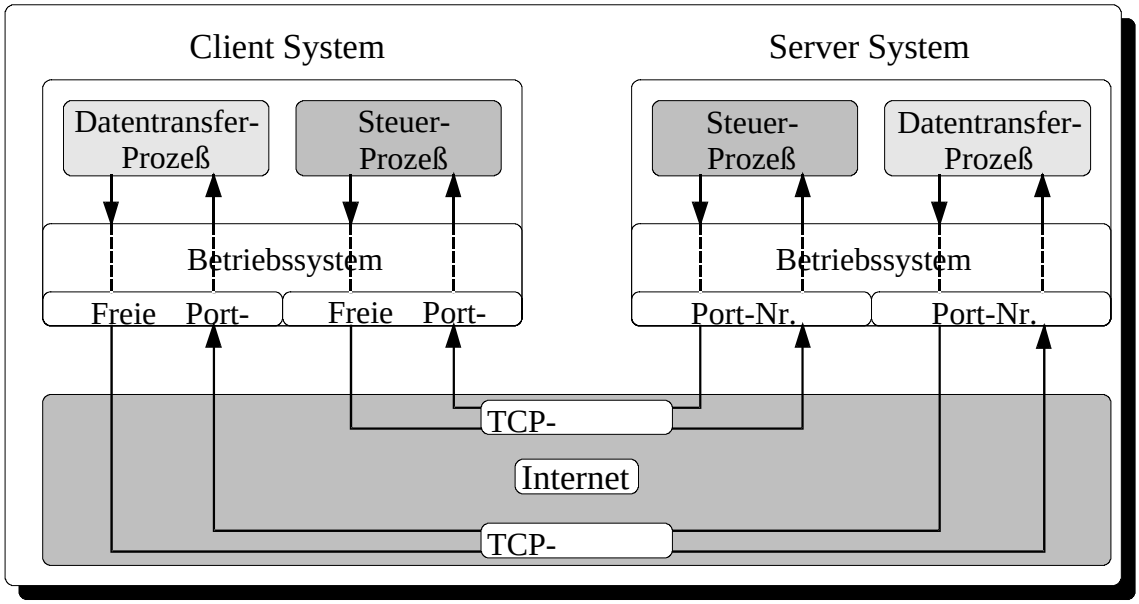



*Bild 4-33 FTP Client-Server Beziehung*

## 4.6.4 TFTP- Trivial File Transfer Protocol

Definiert in RFC 1350 (Juli 1992 Ver. 2)

Das normale File Transfer Protokoll der TCP/IP-Protokollfamilie, FTP, ist vergleichsweise komplex und verlangt sogar auf der einfacheren Client-Seite, dass parallel mehrere TCP-Verbindungen unterhalten werden. Dies ist auf kleineren Systemen ohne adäquate Betriebssystemunterstützung nicht leicht zu realisieren ist.

Für solche Anwendungsfälle, wo die volle Funktionalität von FTP nicht erforderlich und die Komplexität nicht erwünscht ist, enthält die TCP/IP-Protokollfamilie ein sehr viel einfacheres File Transfer Protokoll, das Trivial File Transfer Protocol - TFTP.

TFTP sieht keine Authentisierung vor und unterstützt nur einfache File Transfers. Es basiert auch nicht auf TCP als Transportprotokoll, sondern auf dem sehr viel einfacheren UDP. Um die Verbindung gegen Fehlerbedingungen (z.B. Verluste von Datagrammen) robust zu machen, existiert symmetrisch auf Sender- und Empfängerseite ein Timeout- und Wiederholungsmechanismus. Ein File wird in Blöcken fester Länge (512 Bytes) transferiert, die mit "1" beginnend durchnumeriert werden. Nach dem Absenden eines Blockes wartet der Sender die Bestätigung der betreffenden Block-Nummer ab. Trifft diese nicht (rechtzeitig) ein, wird der Timeout also wirksam, so wird der Block wiederholt. Umgekehrt bestätigt der Empfänger einen empfangenen Block sofort mit seiner Block-Nummer. Trifft vor Ablauf des

Timers kein weiterer Block ein (obwohl der File noch nicht komplett übertragen ist), so wird die Bestätigung der betreffenden Block-Nummer wiederholt. Das Ende der Übertragung wird durch einen Block mit einer Blocklänge von weniger als 512 Byte erkannt.

TFTP beschränkt sich auf wenig Kernfunktionen, und die Implementationen nehmen deshalb auch vergleichsweise wenig Speicherplatz ein, so dass das Programm auch leicht in einem ROM untergebracht werden kann.

## 4.6.5 DHCP - Dynamic Host Configuration Protocol

Definiert in RFC 2131 (März 1997) – Dynamic Host Configuration Protocol

RFC 2132 (März 1997) – DHCP Options and BOOTP Extensions

Ein Nachteil des IP-Protokolls im Vergleich zu IPX oder NetBEUI liegt in der Notwendigkeit, eine Vielzahl von Parametern (IP-Adresse, Subnetzmaske, ...) von Hand konfigurieren zu müssen. Dies ist zum einen lästig, zum anderen aber auch fehlerträchtig. Um dies zu lösen, wurde eine Protokoll zur automatischen Konfiguration entwickelt. Genau genommen existieren derzeit zwei Protokolle: das BOOTstrap Protocol (BOOTP) und sein Nachfolger, das Dynamic Host Configuration Protocol (DHCP). Beide Protokolle verwenden UDP zum Transport ihrer Nachrichten. Im Gegensatz zu RARP können mit diesen Protokollen neben der eigenen IP-Adresse eine Vielzahl weitere Parameter von einem Server bezogen werden. Beide Protokolle sind sich sehr ähnlich, daher wird zunächst die Funktionsweise von BOOTP behandelt, danach werden die mit DHCP eingeführten Erweiterungen besprochen.

### 4.6.5.1 BOOTP

Um seine Konfigurationsdaten zu erhalten, sendet ein Knoten ein UDP-Paket als Broadcast an den für BOOTP spezifizierten Port (UDP Port 67). Der BOOTP-Server empfängt dieses Paket und antwortet mit einem Paket an die Broadcastadresse, in dem die notwendigen Konfigurationsdaten stehen. Der BOOTP-Server kann kein IP-Paket an den Rechner direkt senden, da der Knoten seine eigene IP-Adresse zu diesem Zeitpunkt noch nicht kennt.

Mit einer BOOTP-Nachricht können die folgende Konfigurationsinformationen übertragen werden:

- IP-Adresse des Clients
- IP-Adresse des Routers
- Name eines Bootimages
- Zusätzliche Informationen wie z.B. IP-Adresse eines Time Servers

Nachteilig an BOOTP ist, dass jedem Knoten eine feste IP-Adresse zugeordnet werden muss. Dies hat einerseits einen relativ hohen Pflegeaufwand zur Folge, da für jeden neuen Rechner eine Konfiguration auf dem BOOTP-Server definiert werden muss. Andererseits ist es damit nicht möglich, dass die Zahl der Rechner die Zahl der möglichen IP-Adressen übersteigt. Dies kann aber z.B. bei einem Unternehmen mit vielen, ständig den Standort wechselnden Mitarbeiter, der Fall sein.

#### 4.6.5.2 DHCP

DHCP erweitert BOOTP in mehreren Punkten. Zum einen werden mit einer DHCP-Nachricht weitere Parameter, wie z.B. die Subnetzmaske, übertragen. Zum anderen bietet DHCP mehrere Möglichkeiten, einem Client eine IP-Adresse zuzuordnen [22]:

- manuelle Konfiguration: eine IP-Adresse wird manuell einem Knoten zugeordnet; dies entspricht der Arbeitsweise von BOOTP
- automatische Konfiguration: ein Knoten erhält bei der ersten Anfrage automatisch eine feste IP-Adresse
- dynamische Konfiguration: jeder Knoten erhält dynamisch eine IP-Adresse

Von den soeben genannten Möglichkeiten stellt die dynamische Konfiguration die interessanteste Neuerung dar. Mit dieser Option ist eine vollständig automatische Konfiguration ohne manuelle Eingriffe möglich. Anders als bei der festen Zuordnung erhält ein Client eine IP-Adresse dabei nur für eine bestimmte Zeit (lease period), die dem Client mit der DHCP-Nachricht mitgeteilt wird.

Die dynamische Konfiguration verläuft wie folgt [22]:

- Wenn der Client bootet, befindet er sich im Status INITIALIZE. Um eine IP-Adresse zu erhalten, sendet er eine DHCPDISCOVER-Nachricht via Broadcast an alle Server im Netz und wechselt in den Status SELECT.
- Alle DHCP-Server im lokalen Netz empfangen dieses Datagramm und antworten darauf mit einer DHCPOFFER-Nachricht.
- Der Client wählt eine der Antworten aus (in der Regel die erste, die er empfängt) und sendet an den Server eine DHCPREQUEST-Nachricht, um die in dem Paket angebotene Konfiguration zu bestätigen und mit dem Server die Gültigkeit zu vereinbaren. Mit dem Senden der Nachricht wechselt der Client in den Status REQUEST.
- Der Server antwortet mit einer DHCPACK-Nachricht, woraufhin der Client nach dem Empfang in den Status BOUND wechselt und beginnt, die Konfiguration zu verwenden.

Wird ein Client heruntergefahren, gibt er die im zugeteilte Konfiguration mit einer DHCPRELEASE-Nachricht wieder frei. Der Server kann daraufhin die Konfiguration einem anderen Client zuordnen.

Wenn der Client in den BOUND Status wechselt, werden drei Timer gestellt. Der erste Timer läuft normalerweise nach der Hälfte der lease period ab und veranlasst den Client, seine Konfiguration zu erneuern. Dazu sendet er erneut eine DHCPREQUEST-Nachricht mit seiner aktuellen Konfiguration an den Server und wechselt in den Status RENEW. Der Server kann entweder diese Anfrage positiv beantworten (sendet DHCPACK, die lease period beginnt wieder) oder ablehnen (DHCPNACK), woraufhin sich der Client in den Status INITIALIZE begibt und versucht, eine neue Konfiguration zu erhalten.

Erhält der Client keine Antwort von „seinem“ Server, nimmt er an, dass der Server nicht in Betrieb ist. Er beginnt nach Ablauf des zweiten Timers (87,5% der lease period) damit, seine DHCPREQUEST-Nachricht via Broadcast zu verschicken und wechselt vom Status RENEW in den Status REBIND. Erhält er daraufhin von einem

Server eine positive Antwort, begibt er sich in den Status BOUND, bei einer negativen Antwort in den Status INITIALIZE.

Sollte auch auf diesen Versuch keine Antwort erfolgen, begibt sich der Client nach Ablauf des dritten Timers (100% der lease period) in den Status INITIALIZE und versucht, eine neue Konfiguration zu erhalten.

## 4.7 IP Version 6

Definiert in RFC 2460 (Dez. 1998) – IPv6 Specification

RFC 2462 (Dez. 1998) – IPv6 Stateless Address Autoconfiguration

Als Nachfolger für IPv4 wurde die Version 6 (IPv6) des IP-Protokolls entwickelt, auch unter dem Namen Internet Protocol next Generation (IPnG) bekannt. IPv5, die erste Weiterentwicklung von IPv4 wurde nur experimentell implementiert und hat keinerlei Verbreitung in kommerziellen Geräten gefunden.

Auffälligste Änderung der Version 6 gegenüber IPv4 ist die drastische Erweiterung des zur Verfügung stehenden Adressraums. Obwohl IPv4 mit einem Adressraum der Größe $2^{32}$ eigentlich ausreichend Adressen (über 4 Milliarden) bereitstellt, ist durch die Einteilung des Adressraumes in verschiedene Adressklassen ein Engpass entstanden. Die Folge ist, dass derzeit nur noch Klasse C-Netze verfügbar sind, so dass große Netze keinen zusammenhängenden Adressraum mehr erwerben können. Durch die Zuteilung mehrerer aufeinanderfolgender Klasse C-Netze sind jedoch mehrere Einträge für ein Ziel notwendig, so dass die Routing-Tabellen komplexer werden. Bei gleichbleibendem Wachstum des Internets und der Einführung neuer Anwendungen wird der verfügbare Adressraum um das Jahr 2000 an seine Grenzen stoßen, so dass hier Handlungsbedarf besteht.

Neben dem begrenzten Adressraum sind weitere Schwachpunkte von IPv4 die unzureichenden Sicherheitsvorkehrungen sowie die Tatsache, dass IPv4 den wachsenden Anforderungen wie Multimedia und Echtzeitanwendungen nicht gewachsen ist. Die wichtigsten Neuerungen von IPv6 werden im folgenden dargestellt.

### 4.7.1 Adressierung

IPv6 verfügt über eine Adresslänge von 16 Octets; dies ergibt einen Adressraum von $2^{128}$ (theoretisch $3{,}4 \times 10^{38}$ Adressen). Damit wären selbst bei sehr uneffizienter Verwendung mehr als 1500 IP-Adressen pro Quadratmeter der Erdoberfläche möglich. Derzeit sind nur etwa 15 Prozent der zur Verfügung stehenden Adressen zur Nutzung vorgesehen. Die restlichen 85 Prozent bleiben für zukünftige Anforderungen reserviert.

Um eine kompakte Darstellung der IPv6-Adressen zu ermöglichen, wurde eine neue Notation eingeführt. IPv6-Adressen werden in Hexadezimaler Form angegeben, wobei jeweils 2 Octets durch einen Doppelpunkt getrennt werden:

1080:0000:0000:0000:0008:0800:417A:FB32

Um die Schreibweise zu vereinfachen dürfen führende Nullen weggelassen werden. Adressteile, die komplett aus Nullen bestehen, werden nur durch aufeinander-

folgende Doppelpunkte repräsentiert. Die obige Adresse könnte also auch folgendermaßen geschrieben werden:

1080::8:800:417A:FB32

Zwei aufeinanderfolgende Doppelpunkte bedeuten, dass der Zwischenraum mit Nullen aufgefüllt wird, bis die Maximallänge von 16 Bytes erreicht ist. Sonderfälle wie etwa 0:0:0:0:0:0:0:1 darf man sogar als ::1 abkürzen.

Soll nur ein Teil einer Adresse spezifiziert werden, z.B. ein Netzpräfix, wird dies wie auch bei IPv4 mit CIDR gekennzeichnet. Die Adresse

3521::1AD0/60

definiert die linken 60 Bit einer IP-Adresse.

Ein wichtiges Ziel von IPv6 ist die Kompatibilität zu IPv4. Man trägt dem dadurch Rechnung, dass IPv4-Adressen in reservierte IPv6-Adresse integriert werden. Die Schreibweise hierfür ist eine Kombination aus bis zu sechs Hexadezimalzahlen und vier Dezimalzahlen.

Die IPv4-Adresse 134.108.56.60 würde beispielsweise als ::134.108.56.60 dargestellt werden, sofern das Interface IPv4 und IPv6 unterstützt (gemischte IPv4 und IPv6-Umgebung). Unterstützt das Interface nur IPv4, wird die Adresse auf die IPv6-Adresse ::FFFF:134.108.56.60 abgebildet und darf nur in Tunneln eingesetzt werden.

Bei IPv6 werden grundsätzlich folgende Arten von Adressen unterschieden [25]:

- Unicast-Adressen, die einem Interface eines Netzknotens zugeordnet sind
- Anycast-Adressen, die zur Adressierung mehrerer Interfaces eines oder mehrerer Netzknoten verwendet werden. Ein Datenpaket muss dabei nicht an alle Netzknoten weitergeleitet werden, damit ist eine Aufteilung der Last möglich, wenn ein Knoten über mehrere Anschlüsse an das Netz verfügt.
- Multicast-Adressen, welche zur Definition von Gruppen verwendet werden. Wird ein Paket an eine Multicast-Adresse gesendet, empfangen alle an der Gruppe teilnehmenden Netzknoten dieses Paket. Durch die Verwendung von Multicasts werden die in IPv4 notwendigen Broadcasts vermieden. So ist es z.B. möglich, über bestimmte vordefinierte Multicast-Adressen alle Router eines lokalen Netzes zu erreichen.

Ebenso wie in IPv4 gibt es einige spezielle Adressen, die für bestimmte Verwendungen definiert wurden [25]:

- Eine Adresse, die nur aus Nullen besteht wird als unspezifizierte Adresse bezeichnet. Sie kann während der Initialisierungsphase verwendet werden, um anzuzeigen, dass noch keine Adresse vergeben wurde. Diese Adresse kann niemals als Zieladresse eines IP-Paketes verwendet werden.
- Die Adresse 0:0:0:0:0:0:0:1 (oder ::1) ist als Loopback-Adresse (Adresse des eigenen Interfaces, bei IPv4 127.0.0.1) definiert.

Ähnlich wie bei IPv4 wurde auch bei den IPv6-Adressen eine Aufteilung in mehrere Bereiche vorgenommen, die anhand der ersten Bits unterscheidbar sind. Der größte

Teil dieser Bereiche ist momentan noch nicht zugeordnet, in der folgende Tabelle sind die bereits zugeordneten Bereiche dargestellt [25]:

| Verwendung | Präfix (binär) |
|---|---|
| reserviert für spezielle Anwendungen | 0000 0000 |
| Abbildung von NSAP-Adressen | 0000 001 |
| Abbildung von IPX-Adressen | 0000 010 |
| zusammenfassbare, global eindeutige Adressen | 001 |
| auf eine Verbindung begrenzte Adressen | 1111 1110 10 |
| auf eine Einrichtung begrenzte Adressen | 1111 1110 11 |
| Multicast-Adressen | 1111 1111 |

*Tabelle 4-3 Derzeit zugeordnete IPv6-Adressbereiche*

Zu Beginn der Normierung war eine geographische Aufteilung der Adressen vorgesehen. Dieses Vorhaben wurde jedoch im weiteren Verlauf zugunsten einer Vergabepraxis aufgegeben, die eine hierarchische Zusammenfassung von Verbindungen zwischen einzelnen Netzen ermöglicht. Damit können Adressen von verschiedenen Internet-Teilnehmern von Routern immer dann automatisch zusammengefasst werden, wenn Sie gemeinsame Wege durchlaufen.

Eine zusammenfassbare, global eindeutige IP-Adresse unterteilt sich in folgende Bereiche [25]:

- TLA: Top-Level-Aggregation, z.B. Superprovider oder zentraler Austauschpunkte und Backbones. Diese Einrichtungen bilden die oberste Ebene des Internets
- NLA: Next-Level-Aggregation, z.B. der jeweilige Internet-Provider
- SLA: Site-Level-Aggregation, Standortübergreifende Adressierung innerhalb einer Einrichtung (Subnetze)
- Interface: der lokale Adressraum innerhalb eines physikalischen Netzes

| 3 | 13 | 8 | 24 | 16 | 64 |
|---|---|---|---|---|---|
| 001 | TLA | res. | NLA | SLA | Interface |

*Bild 4-34 Zusammensetzung einer global eindeutigen IPv6-Adresse*

Die komplette Adresse eines Knotens besteht aus der Kombination der lokalen Adresse mit dem Präfix. Das Präfix ergibt sich aus dem durch den Internet-Provider zugeteilten Adressraum.

Die Zusammensetzung einer solchen Adresse soll an einem Beispiel verdeutlicht werden. Ein Firmennetz sei über zwei Internet-Provider (IP1 und IP2) an das Internet angeschlossen, diese Internet-Provider wiederum sind über zwei verschiedene Transit Provider (TP1 und TP2) an das zentrale Netz angeschlossen:

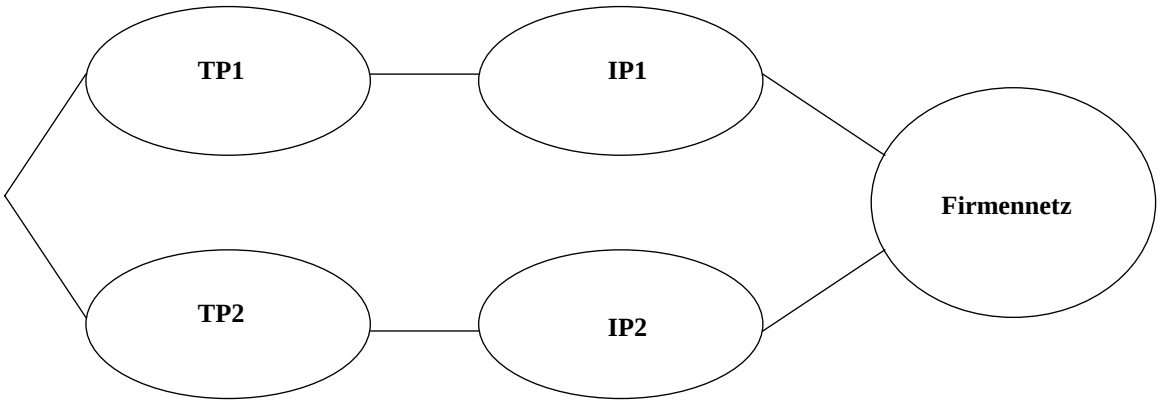



*Bild 4-35 Anschluss eines Firmennetzes über zwei Provider*

Der Internet-Provider IP1 erhält von seinem Transit Provider (TLA=2ABC) den Bereich 00BA:0000 bis 00BA:0FFF. IP2 erhält von TP2 (TLA=2EEE) den NLA-Bereich 00CD:0300 bis 00CD:08FF.

Die Internet-Provider wiederum vergeben einen Teil an unser Netz: IP1 vergibt an uns die Bezeichnung 2B, IP2 vergibt 18E. Den Knoten in unserem Netz stehen jetzt also zwei verschieden Präfixe zur Verfügung:

Zugang über IP1: 2ABC:00BA:002B

Zugang über IP2: 2EEE:00CD:018E

Wenn wir unser Netz noch in weitere Subnetze unterteilen, erhält ein Knoten mit der lokalen Adresse 1234:5678:9ABC:DEF0 im ersten Subnetz (SLA=0001) zwei Adressen, unter der er global im Internet zu erreichen ist.

2ABC:00BA:002B:0001:1234:5678:9ABC:DEF0

2EEE:00CD:018E:0001:1234:5678:9ABC:DEF0

In IPv6 ist von Anfang an eine automatische Zuteilung von Adressen vorgesehen, so dass eine manuelle Konfiguration zwar möglich, jedoch nicht notwendig ist. Für die automatische Vergabe von Adressen sind zwei Verfahren vorgesehen [25]:

1.) Erzeugung einer lokalen Adresse und Überprüfung der Eindeutigkeit durch ICMP, die sogenannte **statusfreie Adressgenerierung**. Diese Methode erfordert keine manuellen Eingriffe und ist vor allem für kleine Netze bzw. Netze mit einfacher Struktur geeignet.

   Die lokal erzeugte Adresse besteht dabei aus einer Kombination von lokal verfügbaren Hardwareinformationen (MAC-Adresse des Knotens) und den von den Routern verteilten Präfix-Informationen, die den Netzteil der Adresse darstellen. Die IP-Adresse besteht also aus zwei Teilen:

   - Präfix: umfasst die ersten 8 Byte der Adresse und wird manuell vergeben
   - Token: umfasst die letzten 8 Byte und wird automatisch ermittelt

2.) Bezug der Adresse und aller weiteren notwendigen Informationen von einem zentralen Server. Diese Methode erfordert mehr Aufwand, ergibt aber auch eine genauere Kontrolle über die vergebenen Adressen und ist auch gut für größere Netze geeignet.

Die Methode der statusfreien Adressgenerierung soll im folgenden etwas näher betrachtet werden:

Bei der statusfreien Adressgenerierung wird die IP-Adresse aus einer Kombination lokaler Hardwareinformationen und der vom Router verteilten Präfix-Information gebildet. Liegt keine Präfix-Information vor, wird eine nur im lokalen Netz gültige Link-local-Adresse gebildet (Präfix: FE80::0).

Sobald ein Knoten an einem Multicast-fähigen Medium angeschaltet wird, bildet er zunächst eine Link-local-Adresse. Diese Adresse wird auf Eindeutigkeit überprüft, indem ein Nachbar-Anfrage (ICMP-Nachricht, Typ 135, Multicast) ausgesendet wird. Erfolgt auf dieses Paket eine Antwort, ist die Adresse nicht eindeutig und der Knoten stoppt den Prozess und wartet auf eine manuelle Eingabe der Adresse. Im nächsten Schritt wird aus den periodisch von den Router ausgesandten Router-Bekanntmachungen (ICMP, Typ 134) das Präfix ermittelt. Der Knoten kann eine Bekanntmachung auch explizit vom Router durch eine Router-Anfrage (ICMP, Typ 133) an die Multicast-Adresse FF02::2 anfordern. Der Knoten überprüft nachdem er die Router-Bekanntmachung empfangen hat zunächst, ob das Bit zur Verwendung eines Servers gesetzt ist. Ist dies der Fall, werden alle weiteren Konfigurationsdaten von einem DHCP-Server angefordert. Ist dies nicht der Fall, werden aus der Router-Bekanntmachung die Präfixe entnommen. Der Router sendet zusammen mit jedem Präfix eine Angabe über die Zeit, wie lange dieses Präfix gültig ist. Dadurch ist es möglich, bei einem Providerwechsel automatisch die neuen Adressen im Netz zu erzeugen.

### 4.7.2 Paketaufbau und Größe bei IPv6

Der Aufbau von IP-Paketen unterscheidet sich im Header, im Gegensatz zu IPv4 wurde bei IPv6 der Header auf das absolute Minimum gekürzt. Der Basisheader enthält nur die für den Transport der Nachricht erforderlichen Angaben. Werden weitere Informationen benötigt, werden diese nicht wie bei IPv4 in einem Optionsfeld variabler Länge untergebracht, sondern durch Verkettung von beliebig vielen zusätzlichen Headern eingefügt. Transitstationen müssen im Normalfall nur den Basisheader auswerten, wodurch eine schnelle Verarbeitung erreicht werden kann.

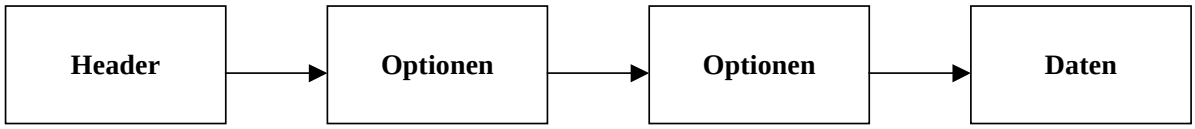



*Bild 4-36 Zusammensetzung des IPv6-Headers*

Der Basisheader hat folgende Aufbau:

| 4 Bit | 8 Bit | 4 Bit | 8 Bit | 8 Bit |
|---|---|---|---|---|
| Version | Class | Flow-Label | | |
| Payload Length | | Next | Hop-Limit | |
| Source Address (128 Bit) | | | | |
| Destination Address (128 Bit) | | | | |

*Bild 4-37 Aufbau des IPv6-Basisheaders*

Im Feld Next wird jeweils der Typ des nächsten Headers eingetragen, so dass sich die Software durch alle Header hangeln kann. Das Feld Class erlaubt die Steuerung des Datenstromes durch Verkehrsklassen. Damit wird festgelegt, mit welcher Priorität Router die Daten auf dem Weg zum Ziel behandeln.

Als neuer Header wurde z.B. der Header zur Fragmentierung definiert. Im Gegensatz zu IPv4 darf ein Paket nicht mehr auf dem Transportweg fragmentiert werden, nur der Absender darf eine Nachricht in mehrere Blöcke aufteilen.

Die minimale Transportgröße von IPv6-Paketen wurde auf 1280 Byte festgelegt (im Gegensatz zu 68 Bytes bei IPv4). Gleichzeitig wurde empfohlen, bei Verbindungen mit variabler Vereinbarung der Transportgröße mindestens den Wert 1500 zu verwenden. Die für den aktuellen Übertragungspfad gültige maximale Blockgröße wird bestimmt, indem der Knoten ein Paket mit der maximalen Blockgröße des angeschlossenen Mediums sendet. Erhält er von einem Router auf dem Übertragungsweg die ICMP-Nachricht „Nachricht zu groß“, so verringert er die Größe auf den im ICMP-Paket angegeben Wert. Dieser Vorgang wiederholt sich, bis das Ziel erreicht wurde.

### 4.7.3 Authentisierung und Verschlüsselung

Definiert in RFC 2401 (Nov. 1998) – Security Architecture for the Internet Protocol

Zentraler Bestandteil von IPv6 ist die Sicherheit. Alle die Sicherheit betreffenden Punkte werden in der IETF-Arbeitsgruppe IPSEC behandelt und die dort entwickelten Standards können ebenso in IPv4 verwendet werden. Die einzelnen Verfahren können in folgende Funktionsbereiche unterteilt werden:

- Authentisierung der Nachricht durch Prüfsummen zum Schutz gegen Veränderung
- Authentisierung des Absenders durch Digitale Signaturen
- Verschlüsselung der Inhaltes zum Schutz gegen unbefugtes Mitlesen

Die Frage der Verteilung und Verwaltung der benötigten Schlüssel wird nicht innerhalb des IPSEC-Standards geregelt, hierfür sind gesonderte Verfahren und Standards zuständig.

Die in IPSEC definierten Standards verwenden die Konstruktion einer Sicherheits-beziehung, mit der ein bestimmtes Verfahren zur Authentisierung oder Verschlüsselung auf einer unidirektionalen Übertragungsstrecke definiert wird. Für eine Kommunikation sind also immer mindestens zwei Sicherheitsbeziehungen

notwendig. Eine Sicherheitsbeziehung definiert folgende Werte und kann durch einen Index angesprochen werden:

- Zieladresse
- Verfahren zur Authentisierung oder Verschlüsselung
- geheimer Schlüssel für das gewählte Verfahren
- weitere, für das Verfahren spezifische Parameter
- Zeitspanne, für die der Schlüssel gültig ist

Innerhalb eines Routers oder eines Endgerätes wird für jede Verbindung mit Hilfe des Indexes auf die zu verwendende Sicherheitsbeziehung verwiesen und dieser Index im Optionsheader eingetragen. Steht im Indexfeld der Wert 0, ist keine Sicherheitsbeziehung vereinbart.

Für die Verschlüsselung stehen zwei verschieden Möglichkeiten zur Verfügung:

- Verschlüsselung der Nutzdaten, der Header bleibt unverschlüsselt. Diese Methode ist vor allem innerhalb eines Netzes von Bedeutung, um sensitive Daten (Passwörter) verschlüsselt zu übertragen
- Verschlüsselung des gesamten Paketes inklusive Header und Transport in einem zweiten IP-Paket (sogenannter Tunnel): mit Hilfe dieser Methode ist es möglich, zwischen Firmenstandorten über des Internet zu kommunizieren, ohne das ein Angreifer aus den Paketen Informationen über die internen Adressen erhalten kann.

Bei allen Verfahren zur Authentisierung oder Verschlüsselung gilt, dass das gesamte Paket verworfen werden muss, wenn kein entsprechender Schlüssel vorliegt. Dieser Vorgang muss in einer lokalen Log-Datei vermerkt werden.

### 4.7.4 Änderungen in ICMP

Definiert in RFC 2463 (Dez. 1998) – ICMPv6 for IPv6

Auch das ICMP-Protokoll hat einige Änderungen erfahren, so dass auch dieses Protokoll nun die Versionsnummer 6 trägt.

Die wesentlichen Änderungen sind:

- Neue Formate für die Übertragung der Adressauflösung, die das Address Resolution Protocol ablösen
- Zusätzliche Elemente zur Definition der MTU wurden eingeführt
- Neue Elemente zur Steuerung vom Multicast-Gruppen ersetzen das bei IPv4 verwendete Protokoll IGMP

ICMPv6-Pakete haben den folgenden Aufbau:

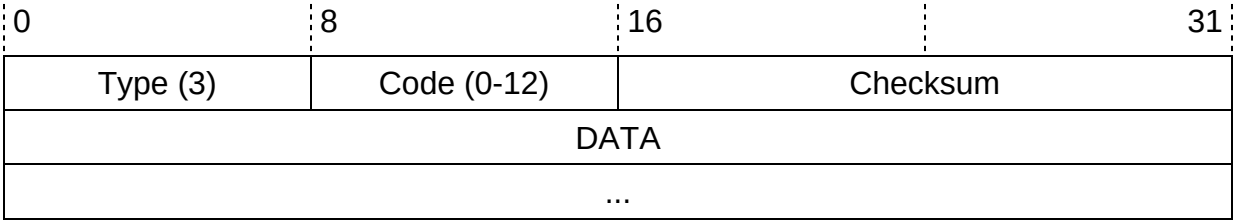



*Bild 4-38 Aufbau eines ICMPv6-Paketes*

Der Inhalt im Datenteil variiert je nach ICMP-Nachricht. Der verwendete Wert im Feld Typ zeigt mit dem höchstwertigen Bit an, ob es sich um eine Fehlernachricht (Bit=0) oder um eine Informations- bzw. Steuernachricht handelt (Bit=1).

Folgende ICMPv6-Nachrichten wurden definiert:

| **Fehlermeldung** | **Typ** |
|---|---|
| Destination Unreachable | 1 |
| Packet Too Big | 2 |
| Time Exceeded | 3 |
| Parameter Problem | 4 |
| **Information, Test und Konfiguration** | **Typ** |
| Echo Request | 128 |
| Echo Reply | 129 |
| Group Membership Query | 130 |
| Group Membership Report | 131 |
| Group Membership Reduction | 132 |
| Router Solicitation (Anfrage an Router) | 133 |
| Router Advertisement (Router-Bekanntmachung) | 134 |
| Neighbor Solicitation (Anfrage an Nachbarn) | 135 |
| Neighbor Advertisements (Bekanntmachung von Nachbarn) | 136 |
| Redirect | 137 |

*Tabelle 4-4 ICMPv6-Nachrichten*

Die letzten 5 ICMP-Nachrichten werden nicht im RFC 2463 definiert, vielmehr sind diese Nachrichten Bestandteil des Protokolls „Neighbor Discovery for IP Version 6” (RFC 2461, Dez. 1998), dass für seine Funktion ICMP-Pakete nutzt.

Das ND-Protokoll löst das ARP-Protokoll für die Abbildung von IP-Adressen auf MAC-Adressen ab. Die im ARP-Protokoll verwendeten Broadcasts werden durch Multicasts ersetzt, bei der die Zahl der potentiellen Empfänger sehr gering sind. Im einzelnen läuft die Adressauflösung wie folgt:

- Aus den unteren 24 Bit der gewünschten IP-Adresse und dem festen Anteil FF02::1:FF00:0/104 wird eine spezielle Multicast-Adresse gebildet, die alle Geräte mit der gesuchten IP-Adresse enthält
- Mit dieser IP-Adresse als Zieladresse wird ein ICMP-Paket vom Typ 135 gebildet und versendet. Dieses Paket wird auch auf MAC-Ebene als Multicast versendet und spricht im Regelfall nur den gewünschten Partner, auf jeden Fall aber nur sehr wenige Rechner an
- die gewünschte Station antwortet mit einem ICMP-Paket vom Typ 136, das die gewünschte MAC-Adresse enthält

### 4.7.5 Notwendige Änderungen an Protokollen höherer Schichten

Die Änderungen in IPv6, insbesondere die Verlängerung der Adressen, macht Anpassungen an vielen Protokollen, die auf IP aufbauen, notwendig. Im folgenden werden beispielhaft einige notwendige Änderungen aufgezeigt [25].

#### 4.7.5.1 UDP

Für den Transport von UDP-Paketen über IPv6 müssen Anpassungen in der Berechnung der Prüfsumme vorgenommen werden, da in dieser Berechnung der Pseudo-Header verwendet wird, der die Adressen des IP-Paketes enthält. Der Algorithmus zur Berechnung der Prüfsumme hat sich ansonsten nicht geändert, im Gegensatz zu IPv4 ist es jedoch nicht mehr gestattet, auf die Prüfsumme zu verzichten und einfach den Wert 0 als Prüfsumme einzutragen.

#### 4.7.5.2 TCP

Ebenso wie bei UDP muss auch bei TCP das Verfahren zur Berechnung der Prüfsumme auf die neuen IP-Header angepasst werden.

#### 4.7.5.3 DNS

Das DNS-Verfahren muss angepasst werden, um 128 Bit Werte zurückzuliefern. Hierbei darf nicht einfach ein ersetzen der bisherigen 32 Bit langen Werte durch 128 Bit lange Werte erfolgen, da dies Auswirkungen auf die bisherigen Implementierungen hätte. Daher muss es möglich sein, gezielt IPv4 oder IPv6-Adressen abzufragen. Um dies zu ermöglichen, wurde ein neuer Datensatztyp für IPv6-Adressen definiert, der bei der Abfrage verwendet wird. Für die Abbildung von Adressen auf Namen (inverse Abfrage) wurde die neue Domäne IP6.INT geschaffen.

## 4.8 Aufgaben

1 a) Welche Schichten gibt es in der TCP/IP-Protokollarchitektur ?

b) Führen Sie eine Zuordnung dieser Schichten zu den Schichten des ISO/OSI-Modells durch.

c) Nennen Sie mindestens 5 Protokolle für die TCP/IP-Schichten und ordnen Sie diese der jeweiligen Schicht zu.

2 Worin unterscheiden sich Class A, B und C Adressen ?

3 Wofür und aus welchem Grund werden Netzmasken eingeführt ?

4 Das TCP-Protokoll arbeitet verbindungsorientiert. Das UDP-Protokoll arbeitet verbindungslos. Kann eine Anwendung mit dem UDP-Protokoll eine verbindungsorientierte Verbindung aufbauen ? Begründen Sie ihre Antwort.

5 Muss eine Applikation, die über das TCP-Protokoll zu mehreren anderen Applikationen Verbindungen haben soll, mehrere eigene Portnummern besitzen oder reicht eine eigene Portnummer ? Erläutern Sie ihre Antwort.

6 Was ist der Unterschied zwischen einer rekursiven und einer nicht rekursiven Abfrage eines Clients an einen Name-Server ?

7 Ein lokales Netz mit Ethernet verwende TCP/IP-Protokolle. Geben Sie an, welche Protokolle verbindungsorientiert und welche verbindungslos arbeiten.

8 Gegeben sei eine diskless Workstation, deren Betriebssystem und IP-Konfiguration auf einem Server gespeichert ist. Warum kann eine solche Workstation nicht von einem Server, der in einem anderen Subnetz liegt, ihr Betriebssystem herunter laden? Mit welchem Protokoll bekommt sie ihre eigene IP-Adresse?

9 Die FHTE betreibt ein Class B-Netz. Könnte die FHTE auf einen eigenen Name-Server verzichten und stattdessen den Name-Server der UNI-Stuttgart (anderes Class-B-Netz) verwenden?

10 Geben Sie die Subnetzmaske für das kleinste mögliche Subnetz an. Wie viele Knoten können dort adressiert werden? Geben Sie dazu ein Beispiel aus dem Adressraum der FHTE an (Subnetzadresse als classless IP sowie alle zulässigen Knotenadressen)

11 Benötigt ein SNMP-fähiger Repeater eine IP-Adresse und eine MAC-Adresse? Begründen Sie Ihre Antwort.

12 Wozu werden bei der Implementierung des IP-Protokolles Timer benötigt?

13 Das Klasse B-Netz 180.2.0.0 soll in 4 Subnetze unterteilt werden.

   a) Geben Sie die hierfür benötigte Netzmaske an.

   b) Geben Sie für jedes der resultierenden Subnetze die Netzadresse und die Broadcastadresse an.

   c) Wie viele Hosts können adressiert werden, wenn in jedem Subnetz ein Router existiert und dieser nicht als Host zählt (Formel genügt, kein Zahlenwert erforderlich) ? Antwort mit Begründung !

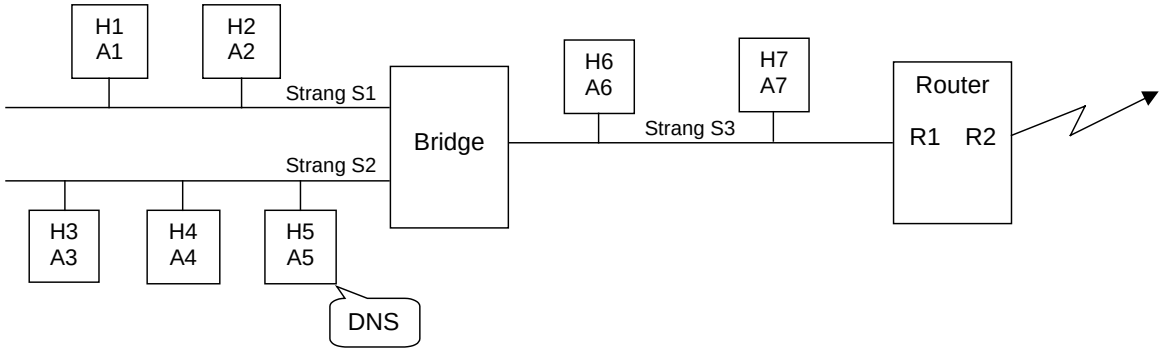
14 Erklären Sie, wie das Hilfsprogramm **traceroute** mit Hilfe von IP-Protokollen den Weg eines Paketes von einem Quellhost zu einem Zielhost ermittelt

15 Welche unbemerkten Fehler können im Rahmen des UDP-Protokolls auftreten, da keine Quittungen verschickt und eingehende Nachrichten auch nicht sortiert werden ?

16 Welche Vorteile bringt die Anwendung der Sliding-Window-Technik beim TCP-Protokoll ?

17 Erklären Sie anhand eines Beispiels, warum es sinnvoll sein kann, applikationsbezogene Aliasnamen für Rechner einzuführen.

18 Gegeben sei das folgende Netz:




H1 bis H7 seien die Hostnamen der Knoten, A1 bis A7 die zugehörigen IP-Adressen. R1 und R2 sind die IP-Adressen des Routers. Die Funktion des Nameservers wird in diesem Netz durch den Rechner H5 erfüllt.

Die Knoten H1 bis H7 haben die MAC-Adressen M1 bis M7. Der Router hat die MAC-Adressen MR1 und MR2.

In dem vorgegebenen Netz arbeiten alle Knoten korrekt. In dem vorliegenden Netz hat bisher noch kein Datenverkehr stattgefunden. Das nachfolgende Kommando ist die erste Kommunikation auf diesem Netz.

Geben Sie bitte auf allen Strängen die Pakete an, die bei dem folgenden Kommando beobachtet werden können. Geben Sie für jedes Paket die MAC- und IP-Adressen, die in diesen Paketen eingetragen sind an. Erläutern Sie bei jedem Paket kurz dessen Funktion.

Folgendes Kommando soll ausgeführt werden: **auf dem Knoten H1: ping H7**

Geben Sie die Pakete in folgender Tabellenstruktur an:

| Paket-Nr | MAC-DA | MAC-SA | IP-DA | IP-SA | Paket-Typ | Erläuterung |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |

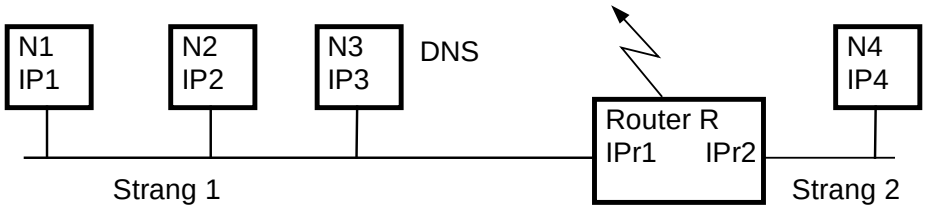
MAC-DA: MAC-Destination Address

MAC-SA: MAC-Source Address

IP-DA: IP-Destination Address

IP-SA: IP-Source Address

19 Gegeben sei folgendes Netz:




N1 bis N4 seien die Knotennamen,

IP1 bis IP4 seien die dazugehörigen Internetadressen

und IPr1 und IPr2 seien die Router-IP-Adressen.

Die Knoten N1 ... N4 sollen die MAC-Adressen M1 ... M4 haben, der Router habe am Eingang mit der Internetadresse IPr1 die MAC-Adresse Mr1 und am Eingang mit der Internetadresse IPr2 die MAC-Adresse Mr2.

N3 erfülle zusätzlich die Funktion des DNS.

Nehmen Sie an, dass im Netz gemäß Bild 1 alle Knoten korrekt arbeiten. Nehmen Sie weiter an, dass alle Knoten an Strang 1 vor folgendem Kommando aus dem vorhergehendem IP-Verkehr alle MAC-Adressen kennen, auf Strang 2 kennen alle Knoten noch nicht (bzw. nicht mehr) die MAC-Adressen der anderen Knoten.

Geben Sie die auf beiden Strängen beobachtbaren Pakete **bis einschließlich** des ersten „ICMP-Echo-Request“ Paketes an, welches auf dem Knoten mit dem Namen „N4“ eintrifft (jeweils mit MAC- und IP-Adressen, IP-Protokolltyp und wesentlichen Inhalt), wenn folgendes Kommando ausgeführt wird:

Auf K1: **ping N4**

Geben Sie die Pakete in derselben Tabellenstruktur wie in Aufgabe 18 an.

# 5 Abstract Syntax Notation One

## 5.1 Einführung

Die Darstellungsebene (Schicht 6) des ISO/OSI-Referenzmodells hat die Aufgabe, die Beschreibung und Kodierung von Informationen in einer gemeinsamen Sprache des offenen Systems zu unterstützen. Während die einzelnen Anwendungsinstanzen lokal über unterschiedliche und systemabhängige Datenrepräsentationen (z.B. ASCII oder EBCDIC) verfügen können, muss beim Datentransfer jeder Kommunikationspartner seine lokale Darstellung der Daten (**lokale Syntax**) in eine von beiden Seiten als gemeinsam akzeptierte Darstellungsform umwandeln.

Diese kodierungsunabhängige Beschreibung der Protokolldateneinheiten (PDU) für eine Anwendung wird als **abstrakte Syntax**, die Sprache zur Spezifikation dieser abstrakten Datenstrukturen und Werte als abstrakte Syntaxnotation bezeichnet. Für die Kommunikation in offenen Systemen wurde eine derartige abstrakte Syntaxnotation mit der Bezeichnung **Abstract Syntax Notation One** (ASN.1) von der ISO als Norm ISO 8824 festgelegt.

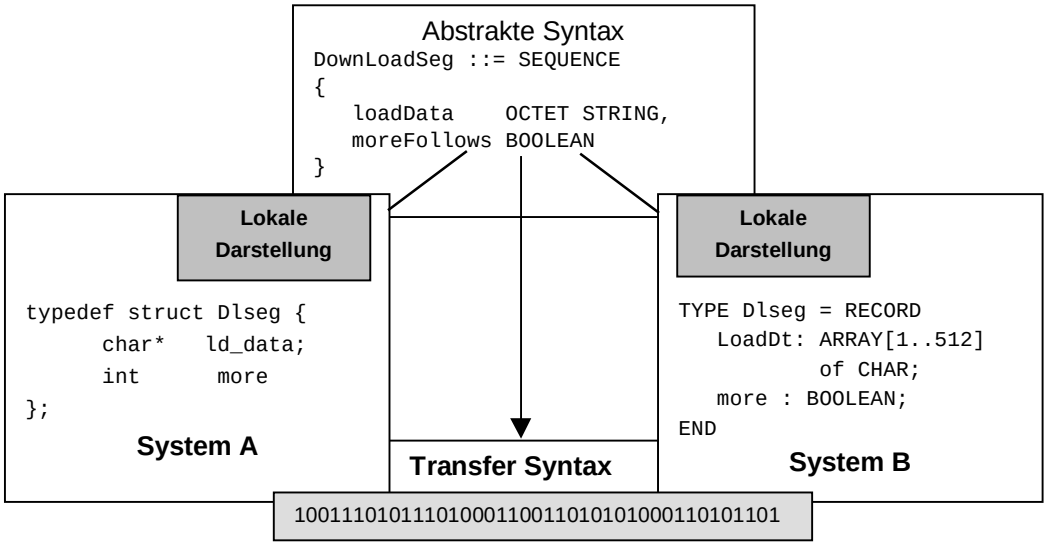



*Bild 5-1 Funktionalität der Darstellungsschicht*

Die Tatsache, dass die Instanzen der verteilten Anwendung von der Syntax der auszutauschenden Daten ein gemeinsames Verständnis haben, reicht zur Kommunikation noch nicht aus.

Erforderlich ist darüber hinaus die Übermittlung der Daten zwischen den Instanzen der Anwendung. Nötig sind also weitere Definitionen, die beschreiben, wie jeder Wert einer abstrakt beschriebenen Datenstruktur bitweise kodiert wird (**Transfersyntax**). Zur Ergänzung der Sprache ASN.1 für die Definition einer abstrakten Syntax wurden im ISO-Standard ISO 8825 Kodierungsregeln (**Basic Encoding Rules**, BER) definiert, durch die die Transfersyntax festgelegt ist.

Das Zusammenspiel zwischen abstrakter, konkreter und lokaler Syntax ist Bild 5-1 verdeutlicht. Beide Systeme verwenden hier unterschiedliche Darstellungsarten, zum Beispiel in Form von Definitionen der Programmiersprachen PASCAL und C.

In dem verbleibenden Teil dieses Abschnittes soll die Syntax dieser abstrakten Beschreibungssprache kurz erläutert werden. Die Kenntnis der Transfersyntax BER ist für die direkte Anwendung von Netzmanagementapplikationen nicht erforderlich. Für eine allgemeine Einführung in BER sei hier auf die Literatur verwiesen.

## 5.2 Verwendung von ASN.1 beim Netzmanagement

Für das Netzmanagement mit SNMP wird ASN.1 für die folgenden zwei Aspekte eingesetzt. Zum einen werden sämtliche Managementinformationen (*Managed Objects*) für das Netzmanagement in ASN.1 beschrieben. Zum anderen dient ASN.1 als Beschreibungssprache für das eigentliche SNMP-Protokoll.

ASN.1 ist eine formale Sprache, sie wird also mit Hilfe einer Grammatik beschrieben. Damit ist es möglich, ASN.1 Compiler zur Verfügung zu stellen, um die Syntax zu überprüfen oder auch Code zu generieren.

Das Internet-Standard Network Management Framework benutzt nur einen Teil der Möglichkeiten von ASN.1. Da eine annähernd komplette Beschreibung von ASN.1 viel zu umfangreich und auch zu komplex ist, soll hier nur der für das SNMP-Netzmanagement wichtige Teil besprochen werden.

### 5.2.1 Module

Eine Sammlung von ASN.1-Beschreibungen, die zu einem gemeinsamen Thema gehören, wird als Modul bezeichnet. Die oberste Syntax für ein Modul ist einfach:

```
<<module>> DEFINITIONS ::= BEGIN
   <<linkage>>
   <<declararions>>
END
```

Der Ausdruck `<<module>>` bezeichnet das Modul sowohl informell als auch möglicherweise zuverlässig. Der Modulname muss zuverlässig (eindeutig) sein, falls das Modul im `<<linkage>>` anderer Module importiert werden soll.

Als Beispiel für das Importieren von Definitionen aus anderen ASN.1 Modulen, sei hier das `OBJECT-TYPE` Makro zur Definition von Managementobjekten erwähnt. Dieses Makro muss von allen MIB's, die Managementinformationen beschreiben folgendermaßen importiert werden:

```
IMPORT OBJECT-TYPE
       FROM RFC-1212;
```

RFC-1212 ist dabei der Name des Moduls, in dem das `OBJECT-TYPE` Makro definiert wurde. Der Ausdruck `<<declarations>>` enthält letztendlich die tatsächlichen ASN.1 Definitionen.

Drei Arten von Objekten sind mit Hilfe von ASN.1 definiert:

- **types:** definieren neue Datenstrukturen und beginnen grundsätzlich mit einem Großbuchstaben z.B. `DisplayString`

- **values:** sind Instanzen (Variablen) eines bestimmten Typs und beginnen grundsätzlich mit einem Kleinbuchstaben z.B. `internet`
- **macros:** werden benutzt, um die Grammatik der ASN.1 Sprache zu erweitern und werden komplett in Großbuchstaben geschrieben z.B. `OBJECT-TYPE.`

Die Schlüsselwörter der ASN.1 Sprache erscheinen grundsätzlich alle in Großbuchstaben.

## 5.2.2 Kommentare

Kommentare werden in ASN.1 mit zwei aufeinanderfolgenden Bindestrichen eingeleitet "`--`". Alle nachfolgenden Anweisungen bis zum Zeilenende werden damit als Kommentar betrachtet.

## 5.2.3 Typen und Werte

Ein neuer ASN.1-Typ wird mit einer einfachen Syntax definiert:

```
NameofType ::= TYPE
```

Ähnlich wird auch der Wert (oder besser eine Instanz eines Datentyps) definiert:

```
nameofValue NameOfType ::= VALUE
```

Das heißt, zuerst wird die Variable mit einem Namen versehen (`nameOfValue`), dann wird ihr Typ festgelegt (`NameOfType`) und schließlich ein Wert zugewiesen. Eine Wertdefinition kann zum Beispiel zur Spezifikation bestimmter, fest vorgeschriebener Werte (Konstanten) benutzt werden:

```
internet    OBJECT-IDENTIFIER ::= { 1 3 6 1 }
mgmt        OBJECT-IDENTIFIER ::= { internet 2 }
mib-2       OBJECT-IDENTIFIER ::= { mgmt 1 }
```

Der Management-Rahmen verwendet vier Arten von ASN.1-Typen, die im folgenden besprochen werden:

### 5.2.3.1 Einfache Typen

Die einfachen Datentypen, die im Management-Rahmen benutzt werden, sind:

**INTEGER:**

Ein Datentyp, der eine natürliche Zahl als Wert hat. Da ASN.1 das Konzept eines Objekts beschreibt, ist keine Bitgenauigkeit definiert. Oftmals ist es praktisch, symbolische Namen für Werte vergeben zu können, z.B.:

```
Status ::= INTEGER { up(1), down(2), testing(3) }
myStatus Status ::= up    -- entspricht myStatus::=1
```

Als Konvention im Management-Rahmen gilt, falls es symbolische Werte für einen ganzzahligen Datentyp gibt, dass nur diese Werte für Instanzen dieses Datentyps gültig sind.

**OCTET STRING:**

Ein Datentyp, der kein oder mehrere Oktette als Wert hat. Jedes Byte in einem Oktett-String kann einen Wert zwischen 0 und 255 annehmen.

**OBJECT IDENTIFIER:**

Ein Datentyp, der die Objekte in der MIB eindeutig identifiziert. Ein `OBJECT-IDENTIFIER` ist eine Folge von nicht negativen ganzzahligen Werten, die einen Baum (Management Information Tree) durchlaufen.

**NULL:**

Ein Datentyp, der als Platzhalter fungiert. Obwohl der Management-Rahmen die Existenz dieses Datentyps zugelassen hat, wird er momentan nicht benutzt.

#### 5.2.3.2 Zusammengesetzte Typen

Die zusammengesetzten ASN.1-Typen, die im Management-Rahmen benutzt werden, sind:

**SEQUENCE:**

Ein Datentyp, der eine geordnete Liste aus anderen ASN.1-Datentypen enthält. Dies entspricht in etwa einer Struktur in der Programmiersprache C.

**SEQUENCE OF :**

Ein Datentyp, der eine geordnete Liste eines ASN.1-Typs enthält. Dies entspricht einem dynamischen ARRAY in den meisten Programmiersprachen. Die Anzahl der Elemente ist dabei normalerweise erst bekannt, wenn das ARRAY erzeugt wird.

**CHOICE:**

Der Datentyp `CHOICE` enthält eine Auswahl von mehreren ASN.1-Typen und entspricht dem `union` Datentyp in der Programmiersprache C.

#### 5.2.3.3 Angehängte Datentypen

ASN.1 bietet zusätzlich noch die Möglichkeit der Definition von neuen Typen durch das „Anhängen" (**tagging**) an einen vorher definierten Typ. Die neuen und alten Typen sind durch unterschiedliche Tags voneinander zu unterscheiden, beziehen sich aber auf denselben Datentyp.

Tags werden zur Beschreibung von ASN.1 Datentypen verwendet. Ein solcher ASN.1 Tag setzt sich aus einer **Tag-Klasse** und einer **Tag-Nummer** zusammen. Bei der Kodierung der abstrakten ASN.1 Syntax in die entsprechende Transfersyntax (BER) wird anstelle des Datentyps die Tag-Klasse und die Tag-Nummer dieses Datentyps übertragen. Sowohl dem Sender als auch dem Empfänger muss die Bedeutung von Tag-Klasse und –Nummer bekannt sein.

ASN.1 definiert die folgenden vier Tag-Klassen:

- **UNIVERSAL**
- **PRIVATE**
- **APPLICATION**
- **CONTEX SPECIFIC**

Von diesen vier Tag-Klassen werden nur die UNIVERSAL und die APPLICATION Tag-Klasse für das Netzmanagement verwendet.

**UNIVERSAL-Tags** sind im ASN.1-Standard vordefiniert und weltweit eindeutig. Beispielsweise wird der ASN.1 Typ **INTEGER** mit **[UNIVERSAL 2]** beschrieben, wobei „2“ die Tag-Nummer darstellt. Ein Ausschnitt aller Universal Tags wird in der folgenden Tabelle dargestellt.

| Tag-Nummer | ASN.1 Typ |
|---|---|
| 1 | BOOLEAN |
| 2 | INTEGER |
| 3 | BIT STRING |
| 4 | OCTET STRING |
| 5 | NULL |
| 6 | OBJECT IDENTIFIER |

*Tabelle 5-1 ASN.1 Universal Tags*

**APPLICATION-Tags** können innerhalb bestimmter Anwendungsklassen bzw. Standards zugewiesen werden und müssen innerhalb dieser Standards eindeutig sein. Für das Internet-Netzmanagement werden neue Datentypen (z.B. Counter oder IpAddress) unter Verwendung der Tag-Klasse APPLICATION definiert.

Die Weitergabe der Tag-Information für die Klassen UNIVERSAL, APPLICATION ist in ASN.1 folgendermaßen definiert:

```
NameOfType ::= [Tag-Class Tag-Number] IMPLICIT TYPE
```

Damit wird ein neuer Datentyp `NameOfType` basierend auf dem bereits definierten Datentyp TYPE erzeugt. Der neue Datentyp besitzt die Tag-Klasse `Tag-Class` und die Tag-Nummer `Tag-Number` und ist somit vom bisherigen Typ `TYPE` unterscheidbar.

Vom Konzept her führt das Anhängen von Datentypen zu einem „Verpacken“ des bestehenden Datentyps, des Tags und Inhalts in einen neuen Datentyp. Das Schlüsselwort `IMPLICIT` ist optional und wird benutzt, falls der neue Datentyp entsprechend den Verpackungsregeln des alten Typs verpackt werden soll.

#### 5.2.3.4 Untertypen

Zur Verfeinerung der Datentypen in ASN.1 können sogenannte Untertypen definiert werden. Von den vielen Möglichkeiten von Untertypen verwendet der Management-Rahmen nur die zwei folgenden:

```
IpAddress ::=   -- in network byte order
     [APPLICATION 0]
          IMPLICIT OCTET STRING ( SIZE (4) )
```

In diesem Beispiel (das auch sehr schön die Verwendung von Tags zeigt) wird ein neuer Datentyp `IpAddress` definiert, der eine Zeichenkette von Oktetten enthält. Über den `SIZE` Operator wird dabei die Länge der Zeichenkette auf genau 4 Zeichen festgelegt.

```
Counter ::=
      [APPLICATION 1]
            IMPLICIT INTEGER (0..4294967295)
```

Hier wird ein neuer ganzzahliger Datentyp definiert dessen Instanzen Werte von 0 bis $2^{32}$ annehmen können. Beide Beispiele sind aus dem RFC1155-SMI entnommen und definieren Datentypen, die speziell für das Netzmanagement von Bedeutung sind.

# 6 Netzmanagement mit SNMP

## 6.1 Grundlagen

### 6.1.1 Die Entwicklung von SNMP

Die Entwicklung des Netzmanagements begann mit **dem Simple Gateway Monitoring Protocol**, das im Jahre 1987 aus der Taufe gehoben wurde. Wie der Protokollname bereits ausdrückt, beschränkte man sich zunächst auf das Management von Gateways.

Aus diesem Protokoll entwickelte sich kurze Zeit später das **Simple Network Management Protocol** (SNMP) zur Verwaltung beliebiger Netzelemente. Und im Mai 1990 wurde SNMP zum Internet Standard erhoben.

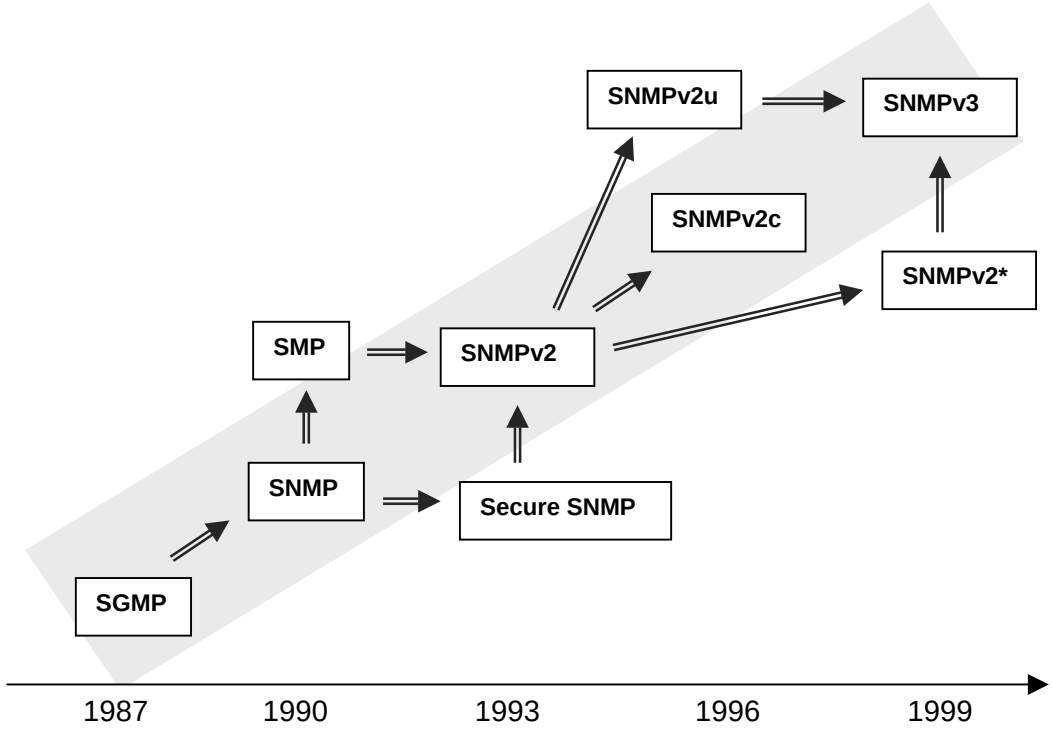



*Bild 6-1 Entwicklung von SNMP*

Aufgrund der einfachen Realisierung des Protokolls, sowie der geringen Anforderungen an die Hardware, fand SNMP rasch eine weite Verbreitung. Durch den umfangreichen praktischen Einsatz kristallisierten sich sehr bald auch die Schwächen von SNMP heraus, insbesondere was das Thema Sicherheit betraf.

Um das Sicherheitsproblem in den Griff zu bekommen, wurden im Juli 1992 eine Reihe von RFCs (allgemein unter dem Namen ***Secure SNMP*** bekannt) *als Proposed Standard* veröffentlicht. Im selben Zeitraum wurde eine Erweiterung von SNMP vorgeschlagen, die die funktionalen Schwachstellen von SNMP lösen sollte. Diese Erweiterungen erhielt die Bezeichnung ***Simple Management Protocol*** (SMP).

Aus beiden Arbeiten von *Secure* SNMP und SMP entstand die offizielle Nachfolgeversion von SNMP: **SNMPv2**. Im März 1993 wurde SNMPv2 zum *Proposed Internet Standard* erhoben. Das Sicherheitskonzept in SNMPv2 wurde jedoch allgemein als zu komplex empfunden, so dass 1996 überarbeitete RFCs zu SNMPv2 veröffentlicht wurden. Diese zweite Version von SNMPv2 wurde allgemein als SNMPv2c bezeichnet und verwendete das bei weitem nicht ausreichende Sicherheitskonzept von SNMPv1.

Um die Sicherheitslücken von SNMPv2 zu schließen, begannen mehrere unabhängige Gruppen damit an einem neuen Sicherheitskonzept zu arbeiten. Daraus kristallisierten sich zwei Ansätze: SNMPv2u und SNMPv2*.

Damit gab es keinen einheitlichen Standard mehr, so dass im Jahre 1997 eine neue Arbeitsgruppe ins Leben gerufen wurde, die sich mit der Entwicklung von **SNMPv3** befassen sollte. Die Arbeiten für SNMPv3 wurden im Januar 1998 als *Proposed Internet Standard* veröffentlicht und im März 1999 zum *Draft Internet Standard* erhoben.

### 6.1.2 Ziele von SNMP

Die Architektur des *Simple Network Management Protocol* basiert auf folgenden Zielen:

- Die Management-Agent Software sollte so **einfach** wie möglich gehalten werden.
- **Remote Management** Funktionen sollten unterstützt werden, um die Vorteile und Möglichkeiten des Internet nutzen zu können.
- Die Architektur sollte so entwickelt werden, dass **Erweiterungen** in der Zukunft einfach durchzuführen sind.
- Die SNMP-Architektur sollte **unabhängig** von speziellen Rechnern und Gateways sein.

Die Idee hinter diesem Konzept ist, durch die Begrenzung von Funktionalität in SNMP letztendlich die Komplexität der Software zu begrenzen. Dies ist wiederum Voraussetzung dafür, dass zum einen die Erweiterbarkeit und Ausbaufähigkeit der Software gegeben ist und zum anderen, dass die meisten Hersteller in der Lage und willens sind SNMP zu unterstützen und für ihre Produkte Agenten zur Verfügung zu stellen.

### 6.1.3 Die Architektur von SNMP

Die Architektur von SNMP basiert auf dem Manager-Agent-Modell, das auch als Grundlage für das OSI Netzmanagement verwendet wurde. Dieses Modell definiert vier Schlüsselelemente für das Management von Netzen:

- Managementstation
- Managementagent
- Management Information Base
- Managementprotokoll

Im folgenden sollen alle vier Elemente näher beschrieben werden:

**Managementstation**

Die **Managementstation** ist in der Regel ein einzelner Rechner, auf dem die Managementapplikation läuft. Die Managementapplikation stellt die Schnittstelle zwischen Netzwerkadministrator und dem Netzmanagementsystem dar. Im einfachsten Fall dient die Managementapplikation zur Aufbereitung und Darstellung der von den Agenten bereitgestellten Informationen. Allerdings können im Manager auch komplexe Managementanwendungen realisiert werden, durch die die Abläufe des Managements automatisiert und der Zustand des Netzes automatisch überwacht werden können.

**Managementagent**

Auf jedem zu verwaltenden Knoten (*Managed Node*) muss ein **Managementagent** installiert sein, der Managementinformationen über diesen Knoten für die Managementstation bereit hält und Anfragen von der Managementstation entsprechend bearbeitet. Ein Agent kann in Ausnahmefällen auch von sich aus Informationen an den Manager senden.

Ein **verwalteter Knoten** kann zum Beispiel ein Rechner, ein Gatewaysystem oder auch ein Netzkoppelelement (Bridge/Router) sein.

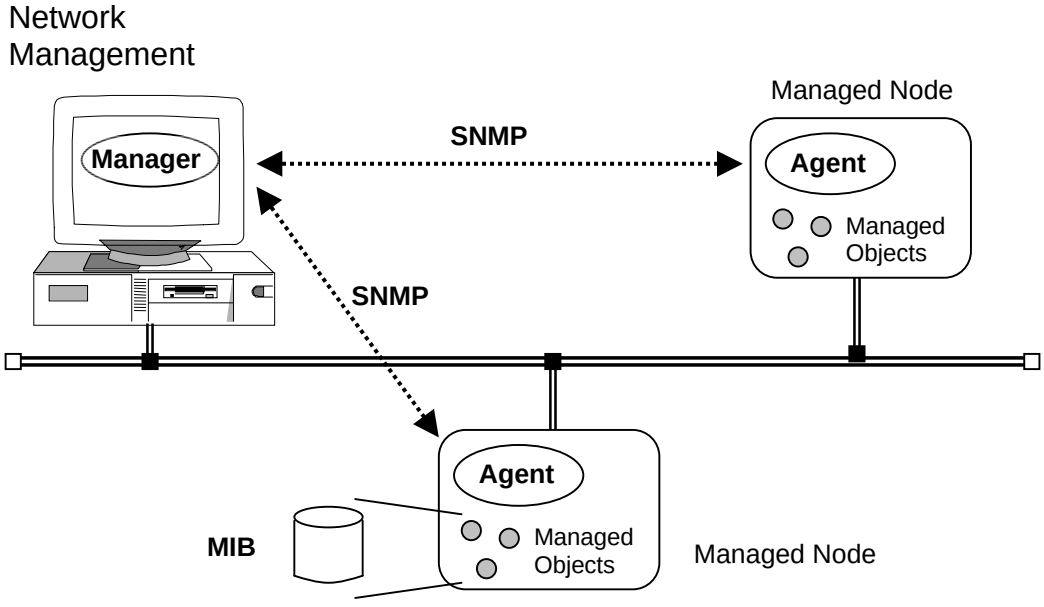



*Bild 6-2 Das Manager-Agent Modell*

Die zu verwaltenden Ressourcen oder Managementinformationen werden im Netzmanagement als Objekte (***Managed Object***) dargestellt. Der Begriff Objekt wurde gewählt, da diese Informationen zu den Eigenschaften einer "normalen" Variablen (wie Name, Typ und Wert) noch weitere Attribute besitzt. Man spricht deshalb auch nicht von Variablen sondern von Instanzen dieses Objekts.

In der Internetwelt wurde auf die Definition komplexer Objekte für das Netzmanagement vollkommen verzichtet, so dass hier der Begriff Variable (genauer **MIB-Variable**) für die Beschreibung einer Managementinformation ebenfalls gebräuchlich ist. Ein Beispiel für ein *Managed Object* ist die Anzahl der gesendeten oder empfangenen Bytes eines Rechners.

**Management Information Base**

Die Ansammlung von *Managed Objects*, die von einem Agenten verwaltet wird, wird unter dem Begriff ***Management Information Base*** (MIB) zusammengefasst. Eine MIB enthält jedoch nicht die eigentlichen Managementinformationen, sondern beschreibt in einer formalen Art und Weise, welche Informationen der Agent bereitstellt und wie die Managementapplikation darauf zugreifen kann.

**Managementprotokoll**

Managementapplikation und Managementagent sind über ein Netzmanagementprotokoll verbunden. Das Protokoll für TCP/IP basierende Rechnernetze ist das ***Simple Network Management Protocol*** (SNMP). Die Architektur des SNMP Protokolls zielt darauf ab, die Anforderungen an die Netzknoten so gering wie möglich zu halten, um Managementfunktionalität auf vielen Netzknoten implementieren zu können. Die Philosophie lautet deshalb:

*Der Einfluss auf verwaltete Knoten durch das Hinzufügen von Netzverwaltung muss möglichst klein sein, also den kleinsten gemeinsamen Nenner darstellen.*

Als Folge dieser Maxime verlagert sich die Last auf die Netzmanagementstation, die den größten Teil der Funktionalität bereitstellen muss.

### 6.1.4 Das Internet-Netzmanagement Rahmenwerk

Der Begriff *Simple Network Management Protocol* bezieht sich eigentlich auf eine Sammlung von Spezifikationen, die in RFCs abgelegt sind und verschiedene Teilaspekte des Netzmanagements abdecken (unter anderem das SNMP Protokoll). Diese Sammlung von Spezifikationen wird auch als das Internet-Netzmanagement Rahmenwerk bezeichnet und ist weiterhin am wachsen.

In den folgenden Abschnitten werden die drei wichtigsten Kerndokumente für das Internet-Netzmanagement vorgestellt. Dies sind:

- RFC1155 / RFC1212 *Structure of Management Information*
- RFC1213 *Management Information Base II (MIB-II)*
- RFC1157 *Simple Network Management Protocol (SNMP)*

Die Dokumente RFC115/RFC1212 „*Structure of Management Information*“ beschreiben Regeln, wie Managementinformationen (*Managed Objects*) innerhalb einer *Management Information Base* beschrieben werden.

Das Dokument „RFC1213 *Management Information Base II*“ definiert allgemeine *Managed Objects* für das Management TCP/IP basierender Netze.

Das Dokument „RFC1157 *Simple Network Management Protocol*“ definiert das eigentliche Managementprotokoll für TCP/IP basierende Netze.

In den folgenden Unterkapiteln werden alle drei Komponenten des Internet-Netzmanagement Rahmenwerks detailliert beschrieben.

## 6.2 Structure of Management Information (SMI)

### 6.2.1 Einführung

Die ***Structure of Management Information*** (RFC1155-SMI) definiert ein allgemeines Rahmenwerk mit Hilfe dessen *Managed Objects* innerhalb einer *Management Information Base* formal beschrieben werden können. Die Definition *der Structure of Management Information* resultiert aus dem ursprünglichen zweigeteilten Ansatz *des* ***Internet Activity Board*** (IAB), des Standardisierungs-Gremiums der Internet-Protokollfamilie:

- Kurzfristig sollte das in der Praxis bereits erprobte **Simple Gateway Monitoring Protocol** (SGMP) für das Netzmanagement erweitert werden. Daraus entwickelte sich das **Simple Network Management Protocol** (SNMP).
- Als langfristige Lösung sollte die Benutzung des sich (immer noch) entwickelnden OSI-Netzverwaltungsprotokolls, dem sogenannten **Common Management Information Protocol** (CMIP), untersucht werden.

Als mittelfristige Lösung wurde vorgeschlagen, das CMIP-Protokoll mit der TCP/IP-Protokollfamilie zu verbinden, woraus das CMOT-Protokoll (**Common Management Information Protocol Over TCP**) hervorging. CMOT ist der Versuch, das ISO/OSI-Management Protokoll auf den verfügbaren IP-Transportmöglichkeiten aufzusetzen. Ansonsten soll sich CMOT auf Applikationsebene wie CMIP verhalten.

Um einen reibungslosen Übergang zwischen den Technologien zu ermöglichen, wurde eine Anzahl von Regeln entwickelt, die eine gemeinsame Basis darstellen sollten. Diese Regeln sind so aufgebaut, dass sie unabhängig von den tatsächlichen Verwaltungsprotokollen verwendet werden können. Sehr früh hat sich jedoch herausgestellt, dass eine für das eine Managementprotokoll definierte MIB nicht ohne weiteres für das andere Managementprotokoll verwendet werden kann. Die bestehenden Schwierigkeiten wurden in RFC 1109 dargelegt und es wurden unterschiedliche Fortentwicklungen von SNMP und CMOT gestattet.

Die SNMP zugrunde liegende Philosophie ist Einfachheit und Erweiterbarkeit zu ermöglichen. So kann eine MIB in SNMP nur einfache Datentypen enthalten, bzw. (2-dimensionale) Tabellen von einfachen Datentypen. Des weiteren unterstützt SNMP im Gegensatz zum OSI Netzmanagement grundsätzlich nur den Zugriff auf einzelne einfache Datentypen.

In den folgenden Abschnitten werden die wichtigsten Regeln für die Beschreibung von Managementinformationen beschrieben.

## 6.2.2 Struktur und Aufbau einer MIB

Die Managementinformationen, die zwischen Manager und Agenten ausgetauscht werden, müssen zunächst in einer *Management Information Base* formal beschrieben werden. Eine MIB ist dabei als hierarchische Baumstruktur aufgebaut, wobei die *Managed Objects* (d.h. die zu verwaltenden Managementinformationen) als Blätter dieser Baumstruktur definiert werden.

Jede MIB ist ihrerseits ein Unterbaum einer noch größeren, ebenfalls hierarchisch aufgebauten Struktur, die in der OSI-Welt als **Management Information Tree** und in der Internetwelt als **Domain Name System** bezeichnet wird. Diese Struktur bildet einen Namensraum, in dem Objekte eindeutig mit Hilfe eines ***Object Identifiers*** angesprochen werden können. Bild 6-3 zeigt einen Ausschnitt aus diesem Baum:

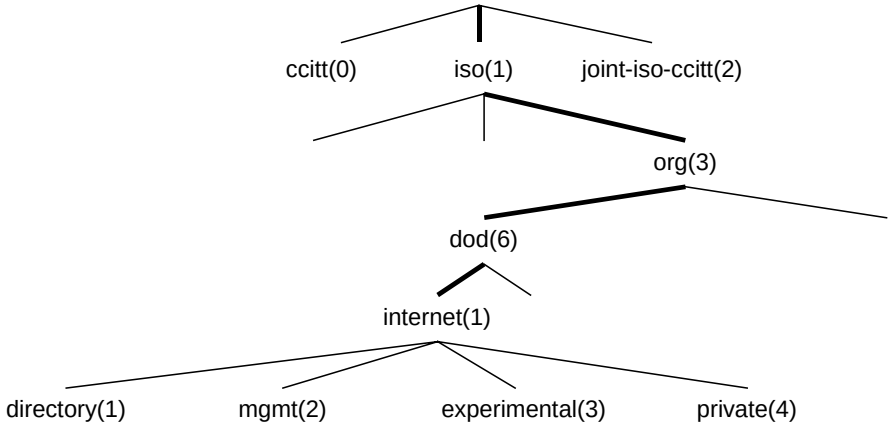



*Bild 6-3 Management Information Tree*

Direkt unterhalb der Wurzel des "*Management Information Tree*" befinden sich drei Zweige: `ccitt (0)`, `iso (1)` und `joint-iso-ccitt (2)`. Diese Teilbäume gliedern sich in weitere Unterbäumen auf, so dass eine baumartige Struktur entsteht. In diesem Baumkonzept können theoretisch beliebig viele Hierarchieebenen vorgesehen werden. Damit ist diese Struktur geeignet, ein ganzes Universum von Objekten aufzunehmen.

Jedes Element einer Hierarchieebene wird durch eine eindeutige Zahl, sowie durch ein Kürzel beschrieben. Der eindeutige Bezeichner eines Objektes, der durch Aneinanderreihen dieser Zahlen oder Kürzel entsteht, wird als ***Object Identifier*** (OID) bezeichnet. Der für das Netzmanagement interessante Teilbaum, der hier etwas dicker hervorgehoben ist, besitzt den *Object Identifier* `1.3.6.1` oder `iso.org.dod.internet`. Auch die gemischte Schreibweise ist üblich: `iso(1).org(3).dod(6).internet(1).`

Die SMI definiert vier weitere Knoten unter dem `internet` Teilbaum:

- Der `directory` Unterbaum ist reserviert für zukünftige Erweiterungen und spielt für das Netzmanagement zunächst keine Rolle.
- Der `mgmt` Unterbaum ist der Teilbaum, der sämtliche vom IAB standardisierte MIBs unter sich hat. Der `mgmt` Teilbaum besitzt nur einen weiteren Unterbaum `mib-2`, der die *SNMP Standard Management Information Base MIB-II* enthält.

- Unterhalb des `experimental` Teilbaums befinden sich neu entwickelte MIBs, die vor ihrer Aufnahme in die Standard-MIB zunächst im `experimental`-Zweig eingeführt und ausgiebig getestet werden. Der Vorteil dieses Ansatzes besteht darin, dass sich Ideen erst als Experimente bestätigen müssen, bevor sie für die Standardisierung in Frage kommen. Falls ein Experiment erfolgreich ist, genießt die entsprechende MIB Vertrauen, wenn ihre Anhänger sie für die Aufnahme in die Internet-Standard-MIB vorschlagen.
- Der `private`-Unterbaum enthält bis jetzt nur eine weitere Verzweigung, den `enterprises`-Unterbaum. Dieser Teilbaum stellt für Unternehmen eine Möglichkeit dar, herstellerspezifische MIBs für ihre jeweiligen Produktfamilien zu entwickeln. Durch die Registrierung bei der Internet ***Assigned Numbers Authority*** (IANA) erhält ein Unternehmen unter dem Baum `internet(1).private(4).enterprise(1)` eine eindeutige Nummer zugewiesen.

  Alle namhaften Unternehmen, die im Netzbereich tätig sind, besitzen einen Unterbaum im `enterprises` Zweig. Für die Festlegung der in diesem Unterbaum spezifizierten Objekte (die in sogenannten **privaten MIBs** beschrieben werden) sind die Firmen selbst verantwortlich.

### 6.2.3 Definition von Objekten

Für Beschreibung der Objekte kommt die OSI Beschreibungssprache "**Abstract Syntax Notation One**" (ASN.1) zum Einsatz. Jedes verwaltete Objekt wird mit Hilfe des `OBJECT-TYPE` Makros beschrieben, das im RFC1155-SMI definiert wurde. Dieses Makro wurde aktualisiert und im RFC-1212 erweitert. Eine etwas vereinfachte Darstellung des `OBJECT-TYPE` Makros ist in der folgenden Abbildung dargestellt:

```
OBJECT-TYPE MACRO ::=
 BEGIN
      TYPE NOTATION ::=
           "SYNTAX"        type(ObjectSyntax)
           "ACCESS"        Access
           "STATUS"        Status
           "DESCRIPTION"   Text | empty
           "REFERENCE"     Text | empty
           "DEFVAL"        "{" value "}" | empty
           "INDEX"         "{" IndexTypes "}" | empty
           VALUE NOTATION ::= value (VALUE ObjectName)
END
```

*Bild 6-4 OBJECT-TYPE Makro aus RFC-1212*

Die Bestandteile des `OBJECT-TYPE` Makros im einzelnen sind:

**SYNTAX**

Die Syntax eines verwalteten Objekts entspricht dem ASN.1 Datentyp des Objektes. Der Ausdruck `ObjectSyntax` steht für einen `CHOICE` Datentyp, der alle im Internet-Netzmanagement definierten Datentypen enthält.

**ACCESS**

Definiert den erlaubten Zugriff auf ein verwaltetes Objekt mit den folgenden Ausprägungen:

- `read-only`: Instanzen des Objekts können nur gelesen werden.
- `read-write`: Instanzen des Objekts können sowohl gelesen als auch gesetzt werden.
- `write-only`: Instanzen des Objekts können gesetzt, aber nicht gelesen werden.
- `not-accessible`: Auf Instanzen des Objekts kann nicht zugegriffen werden.

**STATUS**

Das Status-Feld definiert eine der folgenden Anforderungen an die Implementierung von verwalteten Objekten:

- `mandatory:` der verwaltete Knoten muss dieses Objekt implementieren.
- `optional:` der verwaltete Knoten kann dieses Objekt implementieren.
- `obsolete:` der verwaltete Knoten sollte dieses Objekt nicht mehr implementieren.
- `deprecated:` das *Managed Object* ist überflüssig, kann aber aus Kompatibilitätsgründen noch implementiert werden.

**INDEXPART**

Das Index-Feld bestimmt, welche Objekte innerhalb einer Tabelle als Schlüssel verwendet werden. Diese Index-Objekte müssen eine Tabellenzeile eindeutig identifizieren, wobei nur einfache Datentypen zugelassen sind.

**REFERPART**

Dieses Feld kann eine textuelle Referenz zu einem *Managed Object* in einer anderen MIB beinhalten.

**DEFVALPART**

In diesem Feld kann optional ein Defaultwert für das *Managed Object* festgelegt werden.

**DESCRPART**

Hier kann optional eine textuelle Beschreibung des *Managed Objects* angegeben werden.

**Name (Wert)**

Als Name wird der **OBJECT IDENTIFIER** angegeben. Durch diesen wird *das Managed Object* innerhalb des *Management Information Trees* eindeutig bestimmt.

Anhand dieses Makros werden sämtliche Variablen der Internet-MIB definiert, wie zum Beispiel das *Managed Object* `sysDescr` aus der MIB-II:

```
sysDescr   OBJECT-TYPE
     SYNTAX     OCTET STRING
     ACCESS     read-only
     STATUS     mandatory
     DESCRIPTION
                “A textual description of ...“
::=  { system 1 }
```

*Bild 6-5 Definition des Objektes sysDescr*

Da die Einträge für `INDEX`, `REFERENCE` und `DEFVAL` leer sind, werden sie für das `sysDescr` Objekt nicht aufgeführt. Der Ausdruck `system` ist eine Konstante vom Typ `OBJECT IDENTIFIER` und besitzt den Wert:

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).
```

### 6.2.4 Definition von Tabellen

Die SMI erlaubt zusätzlich zu den einfachen Datentypen auch die Verwendung von zweidimensionalen Tabellen, die ihrerseits wieder nur einfache Datentypen enthalten dürfen. Für die Definition von Tabellen werden die ASN.1 Typen `SEQUENCE` und `SEQUENCE OF` verwendet.

Bei der Definition von Tabellen legt die SMI folgende Konventionen fest:

Für jede Tabelle wird ein Tabellenobjekt mit Hilfe des OBJECT-TYPE Makros definiert:

```
tabellenObjekt OBJECT-TYPE
SYNTAX     SEQUENCE OF zeilenObjekt
ACCESS     not-accessible
...
::= { any-mib 1 }
```

Das Zeilenobjekt wird unterhalb des Tabellenobjekts definiert als:

```
zeilenObjekt OBJECT-TYPE
SYNTAX     ZeilenObjekt
ACCESS     not-accessible
INDEX      spaltenObjekt1..
::=  { tabellenObjekt 1 }
```

Der Datentyp „`ZeilenObjekt`“ wird dabei definiert als:

```
ZeilenObjekt  ::= SEQUENCE {      <Typ von Spaltenobjekt1>
                                  <Typ von SpaltenobjektN> }
```

Man beachte hierbei, dass `ZeilenObjekt` ein Datentyp darstellt (deshalb die Großschreibweise) und `zeilenObjekt` ein *Managed Object*. Die Verwendung desselben Bezeichners für ein Zeilenobjekt und dessen Syntax ist hierbei nicht zwingend vorgeschrieben aber gängige Praxis.

Anschließend werden alle Spaltenobjekte der Tabelle nacheinander mit Hilfe des OBJECT-TYPE Makros definiert:

```
spaltenObjekt1 OBJECT-TYPE
SYNTAX   <Typ von Spaltenobjekt 1>
...
::= { zeilenObjekt 1 }
```

Sowohl das Tabellen- wie auch das Zeilenobjekt sind als not-accessible gekennzeichnet, so dass auf diese Objekte nicht direkt zugegriffen werden darf. Bei der Definition eines Zeilenobjektes wird über das INDEX Feld definiert, welche(s) Spaltenobjekt(e) eine Tabellenzeile eindeutig identifizieren.

### 6.2.5 Datentypen für das Netzmanagement

Jedem Objekt innerhalb einer MIB ist ein Datentyp zugeordnet. Um das Netzmanagement einfach zu halten erlaubt die SMI nur die Verwendung eines Subsets von ASN.1 Datentypen. Dies sind:

```
INTEGER, OCTET STRING, OBJECT IDENTIFIER, SEQUENCE,  SEQUENCE OF.
```

Basierend auf diesen Datentypen werden in der SMI sechs neue Typen für das Netzmanagement definiert:

**IpAddress:** Ein Datentyp, der eine IP-Adresse darstellt. Der Typ IPAddress ist als Untertyp definiert.

```
IpAddress ::=         -- in network byte order
     [ APPLICATION 0 ]
          IMPLICIT OCTET STRING ( SIZE (4))
```

**NetworkAddress:** Ein Datentyp, der die Netzwerkadresse darstellt. Diese kann von möglicherweise mehreren Protokollfamilien stammen. Momentan ist nur die Internetadresse als mögliche Netzadresse eingetragen.

```
NetworkAddress ::=
     CHOICE {
          internet    IpAddress
     }
```

**Counter:** Ein Datentyp, der eine nicht negative ganze Zahl darstellt, die monoton wächst, bis sie ihren höchsten Wert erreicht und dann wieder auf Null springt.

```
Counter ::=
     [ APPLICATION 1 ]
          IMPLICIT INTEGER (0..4294967295)
```

**Gauge:** Ein Datentyp, der eine nicht negative ganze Zahl darstellt, die sowohl größer als auch kleiner werden kann, die aber bei einem bestimmten größten Wert festgehalten wird.

```
Gauge ::=
     [ APPLICATION 2 ]
          IMPLICIT INTEGER (0..4294967295)
```

**TimeTicks:** Ein Datentyp, der eine nicht negative Zahl darstellt, der die Zeit in $^1/_{100}$ Sekunden seit einem früheren Ereignis zählt, allerdings auf die maximale Größe von $2^{32}$ beschränkt ist.

```
TimeTicks ::=
     [ APPLICATION 3 ]
          IMPLICIT INTEGER (0..4294967295)
```

**Opaque:** Ein Datentyp, der eine beliebige Verpackung darstellt.

```
Opaque ::=
     [ APPLICATION 4 ]
          IMPLICIT OCTET STRING
```

Dieser Typ wird als Fluchtmechanismus benutzt, um die Einschränkungen für Datentypen in der SMI zu umgehen.

## 6.3 Management Information Base

### 6.3.1 Einführung

Um die Anforderungen an die verwalteten Knoten so gering wie möglich zu halten, beschränkt sich das Internet-Netzmanagement auf die Abfrage allgemeiner, standardisierter Variablen, den ***Managed Objects***. Diese Managementobjekte werden **formal** in einer ***Management Information Base*** mit Hilfe des `OBJECT-TYPE` Makros beschrieben. Eine MIB stellt somit eine Zusammenfassung sämtlicher Managementinformationen dar, die zwischen Manager und Agent ausgetauscht werden können.

Anhand der MIB besitzt der Manager sämtliche Informationen, die er für das Management eines Netzknotens benötigt. Ein Manager kennt je nach den zu verwaltenden Netzknoten eine Vielzahl von unterschiedlichen MIBs. Auf der anderen Seite kann ein Agent auch mehrere MIBs implementieren. In der Regel wird ein Agent die *Standard Internet MIB* (MIB-II) oder Teile daraus implementieren und zusätzlich herstellerspezifische Informationen in einer privaten MIB zur Verfügung stellen.

Häufig wird eine MIB auch als **verteilte, virtuelle Datenbank** bezeichnet. Die Betonung liegt hierbei auf virtuell, da eine MIB selbst keine Managementinformationen enthält (wie in einer real existierenden Datenbank), sondern diese nur im Sinne von Syntax, Status und Zugriffsmodus beschreibt. Es ist allein die Aufgabe der Agenten, die eigentlichen Managementinformationen zu ermitteln.

## 6.3.2 Die Standard Management Information Base (MIB-II)

Standard Internet MIBs werden unter dem `mgmt`-Unterbaum definiert Unter diesem ist bisher nur ein weiterer Unterbaum, nämlich der `mib-2(1)`-Unterbaum definiert.

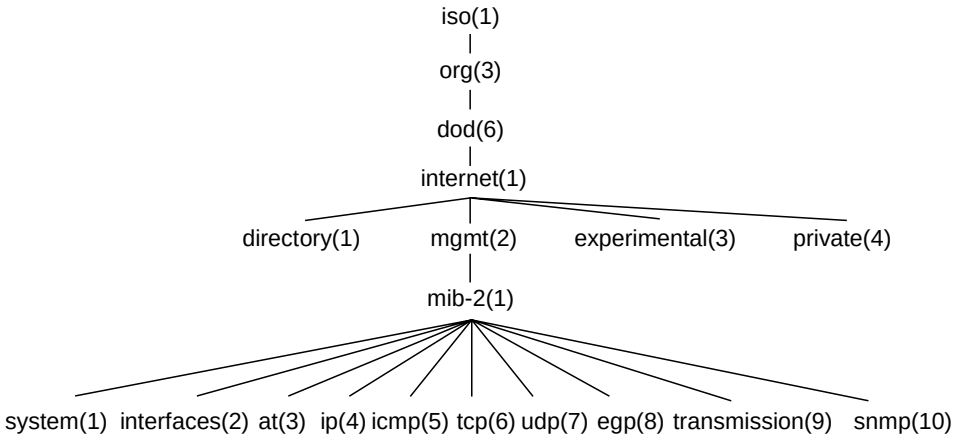



*Bild 6-6 `mgmt`-Unterbaum*

Dieser Teilbaum enthält alle Objekte, die in der *Standard Management Information Base* definiert wurden. Zunächst wurde von dem **Internet Activity Board** (IAB), in RFC1156 die MIB-I definiert. Diese MIB wurde später in RFC1213 zur **MIB-II** erweitert.

In der *Standard Management Information Base* wurden grundlegende Objekte für das Management TCP/IP basierender Rechnernetze definiert. Wenn ein Hersteller von seinem Produkt behauptet, dass es MIB-II fähig ist, dann muss dieses Gerät sämtliche in der Standard-MIB definierte Objekte kennen und bei Anfragen einer Managementstation entsprechende Informationen liefern können.

Ausnahmen gibt es bei Netzknoten, die nicht alle Schichten des TCP/IP Protokollstacks implementieren. So ist zum Beispiel die Abfrage eines Repeaters nach Informationen über die Transportprotokolle TCP oder UDP wenig sinnvoll. Wird ein Objekt nicht unterstützt, so darf die Abfrage dieses Objektes jedoch nicht zu einem Fehler führen. Allgemein gilt die Regel, dass ein Knoten, wenn er Funktionen einer Gruppe enthält, alle Objekte dieser Gruppe unterstützen muss.

### 6.3.2.1 Gruppen der MIB-II

Die Standard Management Information Base II (wie sie in RFC1213 beschrieben wird) besteht aus 10 Gruppen mit insgesamt 171 Objekten. Die Gruppen in der MIB-II reflektieren im wesentlichen den TCP/IP Protokollstack, indem für jede Protokollschicht Managementinformationen definiert wurden.

Im folgenden sollen die einzelnen Gruppen mit ihren Objekten kurz beschrieben werden. Aufgrund der großen Anzahl an Objekten wird an dieser Stelle nur anhand der ersten Gruppe `system` näher auf den Aufbau der Gruppen eingegangen.

### System Gruppe

Die erste Gruppe mit dem Namen `system` beinhaltet allgemeine Informationen zur Konfiguration, wie zum Beispiel eine Beschreibung des Gerätes, der Name und Aufstellungsort des Gerätes usw. Die Gruppe `system` muss von allen verwalteten Knoten implementiert werden.

- `sysDescr (1)`: textuelle Beschreibung des Knotens
- `sysObjectID (2)`: *Object Identifier*, der den Agenten eindeutig identifiziert
- `sysUpTime (3)`: Zeit, die seit dem letzten Booten des Gerätes vergangen ist
- `sysContact (4)`: Name der Kontaktperson
- `sysName (5)`: Gerätename des Knotens
- `sysLocation (6)`: Standort des Knotens
- `sysServices (7)`: Protokollschicht, die der Knoten implementiert

Ein Beispielobjekt aus der System Gruppe sieht folgendermaßen aus:

```
sysDescr          4BSD/ISODE SNMP
sysObjectID       1.3.6.1.4.1.4.1.2.5
sysUpTime         1150006414 (133 Tage, 2:27:44.14)
sysContact        Juergen Kerler
sysName           rhds01.rz.fht-esslingen.de
```

### Interfaces Gruppe

Die Schnittstellengruppe enthält Grundinformationen über die Einheiten in der Schnittstellenschicht. Diese Gruppe enthält in der ersten Ebene zwei Objekte: die Zahl der Schnittstellenanschlüsse des Knotens und eine Tabelle mit Informationen über die Schnittstellen, wie z.B. die Anzahl der gesendeten und empfangenen Bytes.

### AT Gruppe (address translation)

Die Adressübersetzungsgruppe besteht aus einer Tabelle für die Abbildung von IP-Adressen auf Ethernet-Adressen. In der MIB-II wurde die Adressübersetzungsgruppe mit ***deprecate*** (d.h. abgelehnt) markiert, da die Informationen über die Adressabbildung in die IP-Gruppe integriert wurden.

### IP Gruppe

Die IP-Gruppe enthält Daten über das IP-Protokoll. Darin enthalten sind mehrere Zähler, z.B. über die Anzahl der weitergeleiteten, der verworfenen oder der erfolgreich fragmentierten Datagramme. Außerdem enthält diese Gruppe noch drei Tabellen: eine IP-Adresstabelle, eine IP-Routing- und eine IP-Adressübersetzungstabelle. Diese Gruppe muss von allen verwalteten Knoten implementiert werden.

### ICMP- Gruppe

In der ICMP-Gruppe sind für jeden Nachrichtentyp innerhalb des ICMP-Protokolls zwei Zähler definiert. Der eine zählt, wie oft dieser Nachrichtentyp von der eigenen IP-Einheit erzeugt wurde; der andere zählt, wie oft dieser Nachrichtentyp empfangen wurde. Diese Gruppe muss von allen verwalteten Knoten implementiert werden.

**TCP Gruppe**

Die TCP-Gruppe enthält Informationen, die das Transportprotokoll TCP betreffen. Dies sind vor allem Zähler, die z.B. die Anzahl der Verbindungen oder die Anzahl der gesendeten und empfangenen Segmente beinhalten. Die TCP-Group enthält aber auch eine Tabelle, in der alle derzeit bestehenden Verbindungen inklusive lokaler und entfernter IP-Adressen hinterlegt sind. Diese Gruppe muss von allen verwalteten Knoten implementiert werden.

**UDP Gruppe**

Die UDP-Gruppe beinhaltet entsprechende Informationen für das verbindungslose Transportprotokoll UDP. Sie enthält mehrere Zähler und eine Tabelle, deren Einträge darüber Aufschluss geben, welche UDP-Ports derzeit in Verwendung sind. Die UDP-Gruppe muss von allen verwalteten Knoten implementiert werden.

**EGP Group**

Die EGP-Gruppe ist neben der Übersetzungsgruppe die einzige Gruppe, die nicht von allen Knoten unterstützt werden muss. Diese Gruppe enthält spezielle Parameter über das *Exterior Gateway Protocol* (EGP) und muss somit nur auf Knoten implementiert werden, die dieses Protokoll unterstützen, also praktisch nur auf Gateways.

**Transmission Group**

Die Übersetzungsgruppe ist eigentlich überhaupt keine Gruppe, sondern ein Teilbaum, der media-abhängige MIBs enthalten soll. Zum Zeitpunkt der Definition der MIB-II war diese Gruppe leer. In der Zwischenzeit wurden MIBs für das Management von Ethernet-Netzen (RFC1643) und Token-Ring-Netzen (RFC1231) definiert.

**SNMP Group**

Als letzte Gruppe gibt es in der MIB-II noch die SNMP-Gruppe, die **Informationen über das Simple Network Management Protocol** beinhaltet. Dies sind beispielsweise Informationen über das Auftreten von Fehlern oder die Anzahl ausgelesener bzw. veränderter MIB-Objekte. Auch diese Gruppe muss von allen Knoten implementiert werden.

#### 6.3.2.2 Definition der MIB-II in ASN.1

Die Definition einer MIB erfolgt ebenso in ASN.1, wie die Definition einer MIB-Variablen. In Bild 6-7 ist als Beispiel der prinzipielle Aufbau der in RFC-1213 definierten MIB-II abgebildet, wobei in den einzelnen Blöcken nur Auszüge der ursprünglichen Einträge belassen wurden.

```
-- ============================
-- == Definition of the MIB-II ==
-- ============================
RFC1213-MIB DEFINITIONS ::= BEGIN

  -- ======================================
  -- 1. Import definitions from other RFCs –
  -- ======================================

  IMPORTS
      mgmt, NetworkAdress, IpAdress, Counter, Gauge, TimeTicks
                        FROM RFC1155-SMI
      OBJECT-TYPE       FROM RFC-1212;

  -- ======================
  -- 2. Registration points
  -- ======================

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

  -- ====================
  -- 3. Groups in MIB-II
  -- ====================

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

  -- ======================================
  -- 4. Managed objects in the system group
  -- ======================================

  sysDescr OBJECT-TYPE
      SYNTAX DisplayString( SIZE (0..255))
      ACCESS read-only
      STATUS mandatory
      DESCRIPTION "A textual description....."
  ::= { system 1 }

  ...

END
```

*Bild 6-7 Definition einer MIB*

Die MIB-II wird als ASN.1 Modul mit dem offiziellen Namen `RFC1213-MIB` definiert:

```
RFC1213-MIB DEFINITIONS ::= BEGIN
      <module definition>
END
```

Innerhalb der Modul-Definition werden mit der `IMPORTS` Anweisung zunächst Datentypen und Instanzen aus den RFCs `RFC1155-SMI` und `RFC-1212` importiert, damit diese verwendet werden können. Dies sind vor allem die im Dokument RFC1155-SMI definierten Datentypen z.B. `IpAddress` oder `Counter` sowie der *Object Identifier* des `mgmt` Teilbaums. Aus RFC-1212 wird zusätzlich das

überarbeitete OBJECT-TYPE Makro zur Definition der *Managed Objects* in der MIB-II importiert.

In dem Abschnitt „2. Registration points“ wird die Einordnung der Gruppe mib-2 in den MIB-Baum vorgenommen. Der *Object Identifier* der Gruppe mgmt wurde bereits in der IMPORTS Sektion eingefügt, womit automatisch der komplette Pfad von der Wurzel des Baumes bis zum mgmt Knoten ist.

Anschließend werden für jede Gruppe der MIB-II die *Object Identifier* vergeben bevor im Abschnitt „4. *Managed Objects*“ jedes der 171 *Managed Objects* anhand des OBJECT-TYPE Makros beschrieben wird. Als Beispiel für die Definition der einzelnen Objekte wurde das erste *Managed Object* sysDescr aus der system Gruppe dargestellt.

#### 6.3.2.3 Erweiterungen der MIB-II

Neben den in RFC1213 beschriebenen Gruppen enthält der mgmt-Unterbaum mittlerweile eine ganze Reihe weiterer Gruppen, wie zum Beispiel MIBs zum Management von Druckern oder Domain Name Services (DNS).

Als derzeit wohl wichtigste Gruppe ist hier die **RMON-Gruppe** (mib-2 16) zu nennen, die in RFC1757 *Remote Network Monitoring MIB* spezifiziert wurde. Inzwischen gibt es bereits eine Erweiterung dieser Gruppe mit der Bezeichnung RMON-II. Die herausragende Bedeutung der RMON-MIB besteht darin, dass sich das Netzmanagement nicht mehr nur auf die Verwaltung einzelner Knoten im Netz bezieht.

Vielmehr ist es nun möglich, jedes beliebige Gerät im Netz, das einen RMON-Agenten enthält, als Probe für einen Netzmonitor zu benutzen. Mit RMON können zum Beispiel Daten über das derzeitige Verkehrsaufkommen an den einzelnen Interfaces oder die Anzahl der registrierten Kollisionen abgefragt werden. Damit ist es möglich, von der Netzmanagementstation aus, Verkehrsdaten aus dem ganzen Netz zu sammeln.

## 6.4 Das Simple Network Management Protocol

### 6.4.1 Kommunikationsmodell

Für die Kommunikation zwischen Agent und Manager sind grundsätzlich zwei Vorgehensweisen denkbar:

**Agent sendet einen Trap**

Im Falle von **ungewöhnlichen Ereignissen**, z.B. beim Abbruch einer Verbindung oder bei der Überschreitung eines Grenzwertes, sendet der verwaltete Knoten einen **Interrupt** (**Trap** genannt) an die Netzmanagementstation. 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, zum Beispiel CPU-Zeit oder Speicher, und dass bei häufigem Auftreten solcher Ereignisse ein zusätzlicher, von der Managementstation nicht kontrollierbarer Netzverkehr, auftritt. Dies kann

unerwünschte Folgen haben, wenn zum Beispiel in einem überlasteten Netz sämtliche Knoten die Überlastung des Netzes der Managementstation mitteilen und dadurch zusätzlichen Netzverkehr verursachen.

**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 Knoten abfragen und in welchen Abständen diese Abfrage erfolgen soll. Ist der Abstand zu kurz, so wird Übertragungskapazität verschenkt; ist er zu groß, kann die Reaktionszeit auf einen Störfall zu groß sein.

**Trap-directed Polling**

Im Internet-Standard Network Management Framework wird das Modell *des **trap-directed polling*** (Traps beeinflussen das Polling) benutzt. Falls ein besonderes 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. Zur Unterstützung von Traps wird deshalb ein **langsames Polling** von der NMS benötigt.

## 6.4.2 Einordnung von SNMP im TCP/IP Schichten-Modell

Das *Simple Network Management Protocol* ist im Internet Schichten-Modell in der **Applikationsschicht** angesiedelt, also direkt oberhalb der Transportschicht. Es ist zwar theoretisch unabhängig vom Transportmechanismus, da aber jede Nachricht einzeln interpretiert wird, benutzt SNMP normalerweise den verbindungslosen Datagrammservice **UDP**. Dies ist nicht zwingend vorgeschrieben, andere Transportprotokolle wie TCP oder OSI COTS (*Connection Oriented Transport Service*) sind ebenfalls vorstellbar.

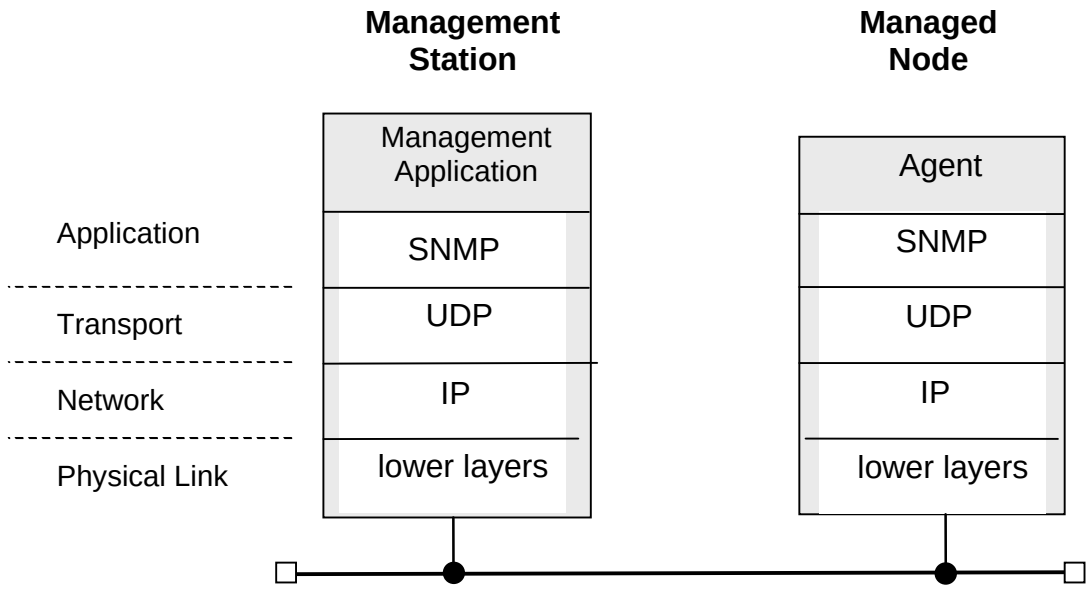



*Bild 6-8 Einordnung von SNMP in das TCP/IP-Schichtenmodell*

Für das UDP-Protokoll spricht, dass ein verbindungsloser Transportdienst am einfachsten zu realisieren ist (Verbindungsaufbau/abbau entfällt) und somit gut in das Konzept von SNMP passt.

**UDP** garantiert jedoch **nicht** die Zustellung von Paketen. Das bedeutet, dass die Managementstation für die korrekte Zustellung der Pakete verantwortlich ist. Dies ist beim Netzmanagement von Vorteil, da die Managementstation selbst den sinnvollen Grad an Paketwiederholungen wählen kann, um sich an schlechte oder überlastete Netze anzupassen.

### 6.4.3 Ausweiskontrolle und Zugriffskontrolle

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, siehe Bild 6-9.

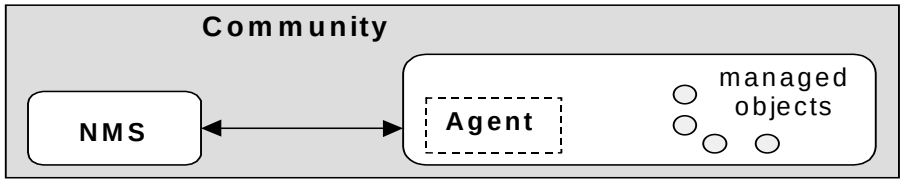



*Bild 6-9 Definition einer Community*

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.

Wenn SNMP-Einheiten Nachrichten austauschen, so enthalten diese grundsätzlich immer drei Anteile:

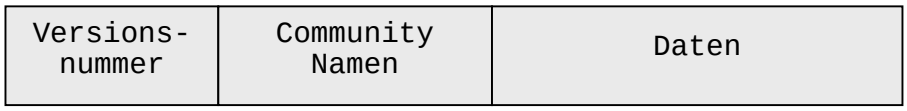



*Bild 6-10 Aufbau einer SNMP-Nachricht*

Das erste Feld jeder SNMP-Nachricht enthält eine **SNMP-Versionsnummer**. Für die erste Version von SNMP ist dieser Wert „0“. Danach folgt der **Community-Name** als Nachweis, dass die sendende SNMP-Einheit Mitglied der angegebenen Community ist und schließlich der **Datenbereich**, der eine SNMP-PDU enthält.

#### 6.4.3.1 Ausweiskontrolle

In SNMPv1 ist nur eine triviale Ausweiskontrolle vorhanden, indem der Community-Namen einfach als Passwort verwendet wird. Erschwerend kommt hinzu, dass der *Community-Name* in jeder SNMP Nachricht unverschlüsselt übertragen wird. Falls dieser 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.

#### 6.4.3.2 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.

Bei der Definition einer MIB wird für jedes Objekt ein Zugriffsrecht festgelegt. Aus der Schnittmenge der Zugriffsmodi für *Managed Objects* und *Community* ergeben sich damit neue Zugriffsrechte für die einzelnen Objekte:

| 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 | — |

*Bild 6-11 Schnittmenge aus SNMP Zugriffsmodus und MIB-Zugriffsmodus*

Mit dem *Community* Konzept ist es damit möglich, dass ein Anwender - je nach verwendeten *Community*-Namen - unterschiedliche Sichten und Rechte für die einzelnen Objekte einer MIB besitzt. So ist es zum Beispiel denkbar, dass nur der Systemverwalter alle MIB-Objekte „sehen“ und falls es der Zugriffsmodus des Objektes erlaubt, auch setzen kann. Für „normale“ Mitarbeiter hingegen ist nur eine Teilmenge der MIB sichtbar und der maximale Zugriffsmodus auf die Objekte auf `read-only` beschränkt.

## 6.4.4 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-Baumstruktur), so ist es zum Beispiel nicht möglich, direkt auf ganze Tabellen oder Zeilen von Tabellen zuzugreifen.

### 6.4.4.1 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`“ angesprochen.

### 6.4.4.2 Zugriff auf Spaltenobjekte

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 *Object 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. Soll auf ein Objekt innerhalb einer Tabelle zugegriffen

werden, so muss der *Object Identifier* des betreffenden Spaltenobjektes und der Wert des Indexobjektes zur Identifikation angegeben werden.

Um beispielsweise in der Schnittstellentabelle `ifTable` die Beschreibung der ersten Schnittstelle zu bekommen, wird der folgende Zugriffsidentifikator verwendet:

```
"1.3.6.1.2.1.2.2.1.2.1"
        Tabellenspalte      | Tabellenzeile
= Object Identifier von ifDescr   = Index der Tabellenzeile
```

SNMP erlaubt auch, dass mehr als ein Spaltenobjekt als Tabellenindex verwendet wird.

## 6.4.5 Operationen von SNMP

Das *Simple Network Management Protocol* ist ein **asynchrones Frage/Antwort Protokoll**. Das heißt, 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 Antworten durch den darunterliegenden Transportdienst verloren gehen können.

Insgesamt stehen vier einfache SNMP Befehle zur Verfügung: `get`, `get-next`, `set` und `trap`. 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
                    }
```

Dieses Feld enthält damit eine Liste von *Managed Objects* repräsentiert durch den jeweiligen *Object Identifier* und dem eigentlichen Wert des Objekts. Im folgenden werden die einzelnen Operationen näher beschrieben.

### 6.4.5.1 Der get Operator

Mit Hilfe des `get` Operators kann der Manager Informationen vom Agenten einholen. Um beispielsweise die Systembeschreibung eines Netzelements 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..">)
```

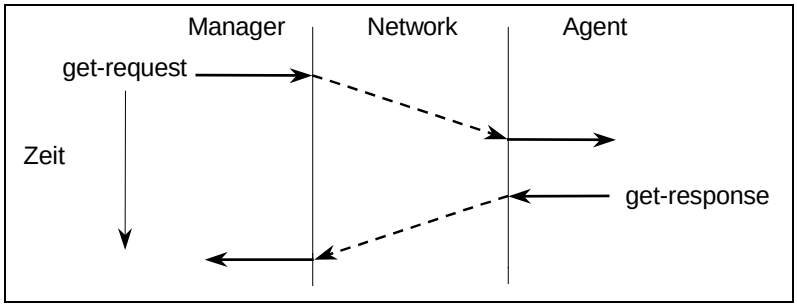



*Bild 6-12 get-Operator*

Der `get` Operator ist **atomar**: Entweder werden die Werte aller abzufragenden Objekte zurückgegeben oder überhaupt keine. Existiert zum Beispiel eines der Objekte in dem `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.

#### 6.4.5.2 Der get-next Operator

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:

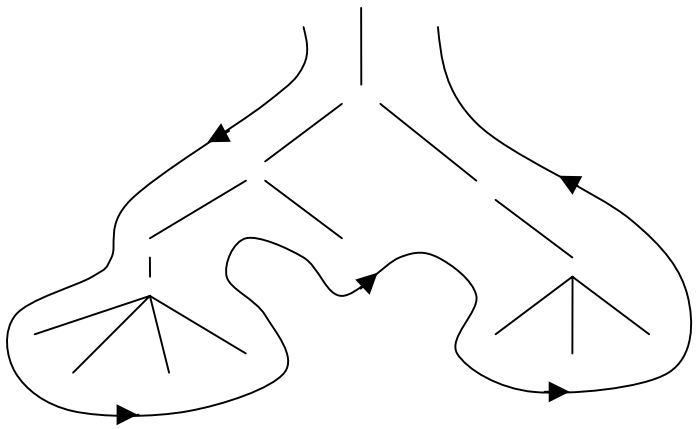


*Bild 6-13 Darstellung der MIB Hierarchie*

Um beispielsweise die Systembeschreibung eines Agenten auszulesen, 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 „`sysDesrc.0`“, so dass der Agent beide Befehle mit der folgenden `GET-RESPONSE PDU` beantwortet:

```
get-response (sysDescr.0, “Router for building...“)
```

Man beachte, dass nicht nur der entsprechende Wert von `sysDescr` zurückgeliefert wird, sondern auch der entsprechende *Object Identifier*! Möchte man nun die

gesamte `system` Gruppe auslesen, so sendet der Manager einfach mehrere `get-next` Befehle, wobei der Wert *des Object Identifiers* vom vorhergehenden Befehl als Eingabe des nächsten Befehls verwendet wird. Der `get-next` Operator eignet sich damit in hervorragender Weise, um Tabellen oder ganze Unterbäume auszulesen.

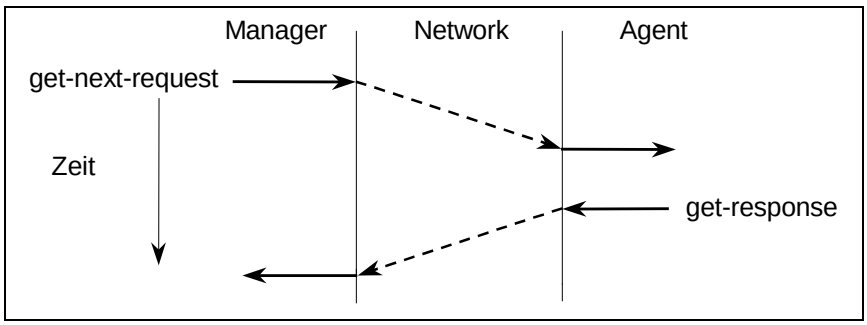



*Bild 6-14 get-next Operator*

Einer der Vorteile des `get-next` Operators gegenüber dem einfacheren `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.

#### 6.4.5.3 Der set Operator

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 folgende `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 ...“>)
```

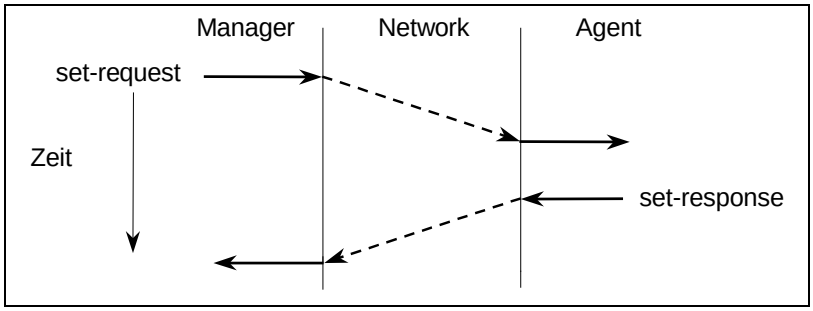



*Bild 6-15 set – Operator*

Der `set` Operator ist ebenfalls **atomar**. Entweder es können alle Werte innerhalb eines Agenten 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`" besitzen.

#### 6.4.5.4 Der trap Operator

Das *Simple Network Management Protocol* ist prinzipiell auf einfaches **Polling** ausgelegt. Das heißt in regelmäßigen Zeitabständen, oder vom Benutzer bestimmt, werden Anfragen an die zu administrierenden Netzkomponenten abgeschickt. Um auf besondere Situationen schneller reagieren zu können, werden sogenannte **Traps** verwendet.

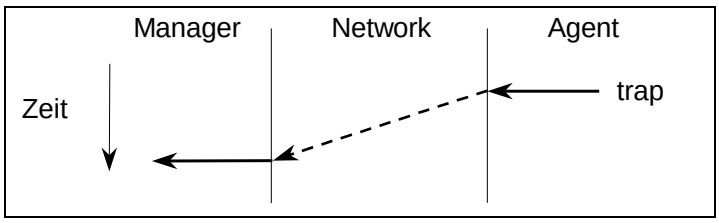



*Bild 6-16 trap-Operator*

Erkennt der Agent eine dieser Situationen, so sucht der Agent, jene Manager heraus, denen er Traps schicken soll. Für jeden Manager wählt er dazu eine passende Community und sendet eine **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. Im Unterschied zu den zuvor geschilderten Operationen erfolgt beim Trap des Agenten keine Bestätigung durch die Managementstation.

## 6.4.6 Aufbau der SNMP-Pakete

Insgesamt wurden für die vier SNMP Operationen nur zwei Protokolldateneinheiten (PDU) definiert: Eine PDU wird für die `get`, `get-next` und `set` Befehle verwendet und die zweite PDU für `trap` Operationen.

### 6.4.6.1 PDU der Befehle GET, GET-NEXT und SET

Für alle drei SNMP Kommandos wird nur eine Protokolldateneinheit benötigt. Damit die SNMP-Empfangseinheiten zwischen den einzelnen Operationen unterscheiden können, für jedes Kommando ein unterschiedlicher ASN1. Tag gesendet.

| request-id | error-status | error-index | varBindList |
|---|---|---|---|

*Bild 6-17 PDU der get, set, get-next Operationen*

Nachfolgend eine kurze Erläuterung der PDU Felder:

**request-id:** 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.

**error-status:** falls der Status nicht Null ist, zeigt er an, dass eine Ausnahme bei der Bearbeitung der Anfrage 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 verändern, eine falsche Syntax oder einen falschen Wert an.

`readOnly`, die gewünschte Operation versuchte, eine Variable zu verändern, die nicht beschrieben werden darf.

**error-index:** ist dieses Feld nicht Null, so zeigt es auf die Variable in der Anfrage, die fehlerhaft war. Dieses 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.

**varBindList**: 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 Agenten mit den entsprechenden Werten aufgefüllt und als `Response`-PDUen an die Managementstation zurückgesendet.

### 6.4.6.2 PDU von Traps

Die Felder einer Trap-PDU werden im folgenden ebenfalls dargestellt und anschließend beschrieben.

| Enterprise | agent-addr | generic-trap | Specific-trap | time-stamp | varBindList |
|---|---|---|---|---|---|

*Bild 6-18 trap-PDU*

**enterprise:** enthält den Wert der MIB-Variablen `sysObjectID` des Agenten und identifiziert die Software des Agenten.

**agent-addr:** enthält die Netzwerkadresse des Agenten.

**generic-trap:** eines von wenigen außerordentlichen Ereignissen:

- `coldstart`, der Agent initialisiert sich (wieder) selbst, und Objekte in seinem View können sich ändern; zum Beispiel Protokolleinheiten auf dem verwalteten Knoten starten neu.
- `warmStart`, der Agent initialisiert sich (wieder), aber die Objekte in seinem View werden nicht verändert.
- `linkdown`, eine angeschlossene Schnittstelle hat ihren Zustand von `up` nach `down` verändert. Die Schnittstelle wird dabei in der ersten Variablen innerhalb des `variable-bindings` Feldes angegeben.
- `linkup`, eine angeschlossene Schnittstelle hat ihren Zustand von `down` nach `up` verändert. Die Schnittstelle wird dabei in der ersten Variablen innerhalb des `variable-bindings` Feldes angegeben.
- `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 Zustand `down` gewechselt. Die erste Variable im `variable-bindings` Feld bezeichnet dabei die IP-Adresse des EGP-Partners.
- `enterpriseSpecific`, ein anderes unerwartetes Ereignis ist eingetreten. Es wird im specific-trap Feld genauer bezeichnet.

**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.

Mit Hilfe des Feldes `enterpriseSpecific(6)` können Hersteller von Agenten eigene Traps definieren, indem sie für jeden spezifischen Trap eine innerhalb der Hersteller-MIB eindeutige ID definieren. Diese ID wird in dem `enterpriseSpecific` Feld mit übertragen.

## 6.4.7 Mängel und Probleme von SNMP

Aufgrund des umfangreichen praktischen Einsatzes von SNMP wurden auch sehr schnell die Mängel und Probleme des SNMP Netzmanagements deutlich. Die wichtigsten Kritikpunkte von SNMP werden in den folgenden Abschnitten behandelt.

### 6.4.7.1 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 (siehe 6.4.6.2). Die Entscheidung, Polling statt Traps zu benutzen, schafft für den praktischen Einsatz von SNMP vor allem bei größeren 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 auftretende Probleme sein könnte.

Ein weiterer Mangel bei SNMP ist die **zentrale Ausrichtung** der Management-Station. Bei SNMP existiert unabhängig von der Größe des Netzes nur eine zentrale Netzmanagement-Station. Eine **Verteilung** oder **Dezentralisierung** der Management-Funktionen wie bei OSI ist nicht vorgesehen. Damit muss eine einzige Station alle Knoten im Netz kontrollieren und überwachen. Dies führt wiederum zu einem erhöhten Verkehrsaufkommen, da zum Beispiel eine Vorauswahl der Managementinformationen durch lokale Manager nicht möglich ist.

### 6.4.7.2 Sicherheit

Sicherheit wurde bei SNMP lange Zeit wie ein Stiefkind behandelt. Es gab zwar seit Mitte 1990 Definitionen zur Sicherheit, diese wurden jedoch nie als RFC veröffentlicht, so dass eine weitere Verbreitung verhindert wurde. Ein Netzmanagementprotokoll ohne wirksamen Sicherheitsmechanismus beinhaltet aber folgende Gefahren:

**Maskerade:** Maskerade ist das Vortäuschen einer authorisierten Identität durch einen nicht authorisierten Benutzer, um Managementfunktionen durchführen zu können.

**Modifikation von SNMP-Paketen:** darunter versteht man das unerlaubte Verändern von Parameterwerten in SNMP-Paketen durch eine Transitstation (z.B. Bridges, Router oder Gateway-Rechner). Dies ermöglicht es unauthorisierten Benutzern, authorisierte SNMP-Befehle für ihre eigenen Zwecke abzuändern.

**Veränderung der Reihenfolge:** SNMP-Daten werden über den ungesicherten Dienst UDP transportiert. Entsprechend können Reihenfolgeveränderungen auftreten, die passiv durch die Dynamik des LAN oder aktiv durch Fremdpersonen verursacht werden können.

**Spionage:** das passive Mitlesen von Managementdaten durch unauthorisierte Benutzer ermöglicht die Beschaffung und Weitergabe von LAN-Parametern.

In SNMP werden diese Gefahren nur sehr unzureichend über die Verwendung des **Community-Names** als eine Art Passwort verhindert. Erschwerend kommt hinzu, dass das Passwort**,** wie auch die Nutzdaten, bei der Übertragung **nicht**

**verschlüsselt** sind. Damit ist es sehr einfach, Pakete unerlaubt abzuhören und nach eigenen Vorstellungen zu modifizieren.

Aus diesen Gründen wurde in vielen SNMP-Agenten die `set`-Operation nicht zur Verfügung gestellt. Durch die weite Verbreitung von SNMP und die inzwischen vorhandenen Management-Umgebungen mit ihren Management-Applikationen, ist diese Funktionalität aber dringend erforderlich. Bedingt durch diesen Umstand sind SNMP-basierende Systeme derzeit sinnvoller für den Bereich des **Monitoring** einsetzbar. Für Netze mit sicherheitskritischen Anwendungen scheidet eine **Remote-Rekonfiguration** damit aus.

#### 6.4.7.3 Fehlende Funktionen in SNMP

SNMP kennt nur vier verschiedene Typen von Operationen: `get`, `get-next`, `set` und `trap`. All diese Operationen können nur gezielt auf **einzelne Objekte** zugreifen; eine **Menge** oder **Klasse** von Daten kann nicht mit einem Zugriff transportiert werden. Dies führt in der Praxis des Netzmanagements, in der durchaus Datenkonstrukte vorkommen (etwa Routing-Tabellen), zu Einschränkungen. Auch hier weiß sich SNMP zunächst einmal durch einen Trick zu helfen, indem der get-next Befehl verwendet wird. Mit diesem Befehl ist es zwar möglich Tabellen auf einfache Weise auszulesen, jedoch muss eine Management-Station für jeden Eintrag in der Liste eine `get-next` Operation absetzen. Bei entsprechend großen Tabellen, wie der Routing-Tabelle, führt dies wiederum zu einem erheblichen Netzverkehr durch das Management-Protokoll selbst.

SNMP Operationen liefern zum Teil **wenig aussagekräftige** Fehlercodes. Wenn eine Management-Station zum Beispiel eine Anfrage an ein Gerät stellt, die nicht von diesem unterstützt wird, so gibt dieses Gerät ein leeres Paket zurück. Nach Erhalt der leeren Botschaft wiederholt der Manager die Anfrage und verschwendet somit Bandbreite, um Informationen zu erhalten, die ein Gerät nicht unterstützt. Darüber hinaus können SNMP-Management-Stationen nicht feststellen, warum ein leeres Paket zurückgeliefert wurde. Gleichgültig, ob dieser Parameter nicht unterstützt wird, oder ob das Gerät ausgefallen ist: in beiden Fällen wird ein leeres Paket übermittelt.

Traps werden in SNMP nicht bestätigt. Damit gibt es keine Rückkopplung, dass wichtige Traps ihr Ziel erreicht haben.

## 6.5 Aufgaben

1 Erklären Sie die Architektur des Netzmanagements mit SNMP anhand einer Skizze **(10 min)**. Tragen Sie in diese Skizze ein:

- eine NMS (Network Management Station) mit einem Network Manager,
- mehrere SNMP-fähige Knoten,
- mehrere nicht SNMP-fähige Knoten,
- Agenten,
- einen Proxy-Agenten,
- managed objects
- und verschiedene MIBs.

2 a) Welche Mechanismen gibt es bei der Kommunikation zwischen Manager und Agent bei SNMP ?

b) Auf welchen Transportdienst setzt SNMP (Version1) auf und wieso ?

3 Ordnen sie die Architektur des Netzmanagements mit SNMP einer der beiden folgenden Klassen zu:

- Hierarchische Architektur
- Peer-to-Peer-Architektur (Gleichberechtigte Partner)

Begründen Sie Ihre Entscheidung

a) Erfolgt in diesem Architekturmodell die Kommunikation ereignisorientiert ?

b) Erläutern Sie, was für eine Art von Informationen in jeder der im Bild von Aufgabe 1 eingezeichneten MIB liegt.

4 Aus welchen Komponenten setzt sich das Internet-Netzmanagement-Rahmenwerk zusammen ?

5 Erklären Sie den Begriff trap-directed polling.

6 Auf welchen Transportdienst setzt SNMP (Version1) auf und wieso ?

7 Wie viele und welche SNMP-Operationen gibt es (SNMPv1) ?

8 Erläutern Sie, welche Vorteile das Netzmanagement mit SNMPv2 gegenüber SNMPv1 in Bezug auf die Netzbelastung bringt !

9 Erläutern Sie, was eine MIB ist.

10 Wozu dient ein Proxy-Agent ?

# 7 Weiterentwicklung von SNMP: SNMPv2

Vier Jahre nach der Einführung von SNMP wurde das Nachfolgeprotokoll SNMPv2 (Version 2) von der *Internet Engineering Task Force* (IETF) aus der Taufe gehoben. SNMPv2 entstand als Antwort auf die Mängelliste zu SNMP, die sich aus den umfangreichen praktischen Einsatz von SNMP ergeben hatte.

Im April 1993 wurden 12 Dokumente (RFC1441-1452) mit zusammen über 400 Seiten veröffentlicht, die von der *Internet Engineering Steering Group* (IESG) zum ***Proposed Standard*** erhoben wurden. Im Dezember 1995 wurden diese RFCs allerdings als „historic“ klassifiziert und die IETF veröffentlichte überarbeitete RFCs (RFC1901-RFC1908) zu SNMPv2:

Die wichtigsten Neuerungen von SNMPv2 beziehen sich auf folgende Themen:

- Verwendung von weiteren Transportprotokollen
- Erweiterung der *Structure of Management Information*
- Erweiterung des eigentlichen SNMP Protokolls
- Erweiterung der MIB-II
- komplette Neugestaltung der Zugriffskontrolle und Sicherheit

Wie bereits erwähnt scheiterte SNMPv2 gerade an der Neugestaltung der Zugriffskontrolle und Sicherheit, was einer der Hauptanstoßpunkte von SNMPv1 war. Die rein funktionalen Erweiterungen von SNMPv2 wurden jedoch komplett auch in der dritten Version SNMPv3 übernommen, so im folgenden auf die wichtigsten Neuerungen von SNMPv2 eingegangen wird.

## 7.1 Verwendung unterschiedlicher Transportdienste

SNMPv1 wurde als Netzmanagement-Protokoll für auf TCP/IP basierende Netze entwickelt. Die IP-Adressierungsart ist in SNMPv1 im Protokoll fest verankert: der einzige mögliche Objekttyp für Adressen ist `IpAddress`, eine 32-Bit Internet Adresse. Dieser Objekttyp ist jedoch beispielsweise für die Darstellung von OSI-Adressen ungeeignet. Eine Verwaltung von Netzen, die auf anderen Protokollen basieren (wie zum Beispiel OSI- oder IPX-Netze), ist damit nicht möglich.

Um eine weitere Verbreitung von SNMP zu ermöglichen und um heterogene Netze komplett über SNMP verwalten zu können, wurden zusätzlich zu UDP/IP **weitere Protokoll-Stacks** definiert, über die SNMP Pakete transportiert werden können. Zu diesem Zweck wurde ein neuer Adresstyp `NsapAddress` eingeführt, der aus bis zu 21 Oktetten besteht und zur Aufnahme einer beliebigen OSI-Adresse gedacht ist. Hiermit wurde auch gleichzeitig die Möglichkeit geschaffen, 128-Bit IP-Adressen darzustellen.

Neben dem üblichen (und auch empfohlenen) *User Datagram Protocol* (UDP) werden folgende Transportprotokolle vorgeschlagen:

- *OSI Connectionless-Mode Network Service* CLNS
- *OSI Connection-Oriented Network Service* COTS
- *Novell Internetwork Packet Exchange* (IPX)

## 7.2 Erweiterungen der SMI

Die Erweiterungen der *Structure of Management Information* in SNMPv2 beziehen sich im wesentlichen auf eine überarbeitete Version des `OBJECT-TYPE` Makros sowie auf die Bearbeitung von Tabellen.

### 7.2.1 Definition von Managed Objects

Verwaltete Objekte werden - wie auch in der ersten Version von SNMP - mit Hilfe eines `OBJECT-TYPE` Makros definiert, das in SNMPv2 folgendermaßen definiert wurde:

```
OBJECT-TYPE   MACRO  ::=  BEGIN
TYPE NOTATION   ::=   "SYNTAX"        Syntax
      "UNITS"         Text | empty
      "MAX-ACCESS"    Access
      "STATUS"        Status
      "DESCRIPTION"   Text
      "REFERENCE"     Text | empty
      "INDEX"         "{„ IndexTypes "}" | empty
      "DEFVAL"        "{" value "}" | empty
VALUE NOTATION  ::=  value (VALUE ObjectName)
```

Die wichtigsten Änderungen werden nachfolgend erläutert:

**Units Feld**

Neu hinzugekommen ist die Möglichkeit im `Units` Feld eine Maßeinheit für ein Objekt zu beschreiben. Diese Erweiterung ist besonders hilfreich für Objekte, die in irgendeiner Art messbare Werte beinhalten, z.B. Zeit oder Temperatur. Die Maßeinheit wird dabei als einfacher Text angegeben.

**MAX-ACCESS**

Das Schlüsselwort zur Beschreibung der Zugriffsrechte eines Objektes wurde um den Präfix „MAX“ erweitert, um damit den maximalen Zugriff zu verdeutlichen und kann die folgenden Werte annehmen:

- `not-accessible` auf das Objekt kann nicht zugegriffen werden
- `accessible-for-notify` das Objekt kann nur in einer Notifikation enthalten sein
- `read-only` auf das Objekt kann nur lesend zugegriffen werden

- `read-write` auf das Objekt kann lesend und schreibend zugegriffen werden
- `read-create` auf das Objekt kann lesend und schreibend zugegriffen und das Objekt kann (innerhalb einer Tabelle) vom Manager erzeugt werden

**STATUS**

Beim Status Feld wurden die möglichen Werte neu festgelegt:

- `current` Objekt ist Teil des aktuellen Standards
- `deprecated` Objekt ist veraltet, kann aber aus Kompatibilitätsgründen noch implementiert sein
- `obsolete` Objekt ist veraltet und sollte nicht mehr implementiert werden

### 7.2.2 Neue Datentypen in SNMPv2

In SNMPv2 wurden zwei neue Datentypen für das Management mit SNMP definiert. Dies sind:

```
Counter64  ::= [APPLICATION 6]  IMPLICIT  INTEGER
                                (0 ..18446744073709551615)
```

und

```
Unsigned32 ::= [APPLICATION 2]  IMPLICIT  INTEGER
                                (0..4294967295)
```

Der Datentyp `Unsigned32` ist dabei nur ein Aliasname für den `Gauge` Datentyp und ist von diesem nicht zu unterscheiden. Des weiteren wurden die Bezeichnungen der Datentypen `Gauge` und `Counter` in `Gauge32` bzw. `Counter32` geändert.

### 7.2.3 Tabellen in SNMPv2

Die Definition von Tabellen wurde komplett von SNMPv1 übernommen. SNMPv2 erweitert die Konvention in RFC1212, um das Erzeugen, Löschen und den Zugriff auf Zeilen einer Tabelle zu ermöglichen. Im wesentlichen können zwei unterschiedliche Arten von Tabellen in SNMPv2 definiert werden:

- Tabellen, die es dem **Manager nicht erlauben** Tabellenzeilen zu erzeugen oder zu löschen. Diese Tabellen werden ausschließlich vom Agenten kontrolliert. Die Objekte dieser Tabellen haben entweder `read-write` oder `read-only` Zugriffsrechte.
- Tabellen, die es dem **Manager erlauben** neue Zeilen zu erzeugen, bzw. Zeilen zu löschen. Objekte innerhalb dieser Tabellen besitzen entweder die Zugriffsrechte `read-create` oder `read-only`. Es ist ebenfalls möglich, dass sowohl der Manager als auch der Agent selbst die Anzahl der Tabellenzeilen variieren kann.

#### 7.2.3.1 Indizierung von Tabellen

SNMPv2 schreibt vor, dass Tabellenzeilen eindeutig über einen Index identifiziert werden können. Der Index einer Tabelle wird in dem `INDEX` Teil des `OBJECT-TYPE` Makros festgelegt und kann auch aus mehreren Spaltenobjekten zusammengesetzt sein. Neu in SNMPv2 ist, dass per Konvention alle Indexobjekte das Zugriffsrecht `not-accessible` besitzen.

#### 7.2.3.2 Erzeugen von Tabellenzeilen

Eines der am heißesten diskutierten Punkte in SNMPv2 war, wie Tabellenzeilen hinzugefügt oder gelöscht werden können. SNMPv1 lässt diesen Punkt völlig offen, was dazu führte, dass für jede MIB ein unterschiedlicher Ansatz gewählt wurde.

In SNMPv2 wurde ein Ansatz standardisiert, der sich im praktischen Einsatz der RMON MIB bewährte. Demzufolge besitzt jede Tabelle, die es dem Manager erlaubt Zeilen hinzuzufügen oder zu löschen, per Konvention ein Statusspaltenobjekt. Dieses Statusobjekt besitzt den Datentyp `RowStatus` und die `Zugriffsrechte` `read-create`. Tabellenzeilen werden neu erzeugt oder gelöscht indem der Wert dieses Statusobjektes über eine `set` Operation verändert wird.

SNMPv2 sieht zwei Möglichkeiten vor, wie Zeilen erzeugt werden können: `CreateAndWait` und `CreateAndGo`.

##### 7.2.3.2.1 CreateAndWait

Bei der `CreateAndWait` Methode erzeugt der Manager eine neue Tabellenzeile, indem er das Statusobjekt mit einem neuen Index auf den Wert „`CreateAndWait`“ setzt. Es liegt in der Verantwortung des Managers einen neuen eindeutigen Index für die erzeugende Tabellenzeile zu wählen. Der Agent erzeugt daraufhin die Zeile mit dem neuen Index und setzt alle Objekte der neuen Zeile mit `read-create` Zugriffsrechten auf ihre Defaultwerte. Anschließend setzt der Agent das Statusobjekt auf den Wert `NotInService`.

Gibt es keine Defaultwerte für ein oder mehrere Objekte der Tabelle, setzt der Agent das Statusobjekt auf den Wert `NotReady` und zeigt damit dem Manager an, dass dieser noch Werte für einzelne Objekte der neuen Zeile setzen muss.

In beiden Fällen kann der Manager über eine `get` Operation Werte und Zustand der neuen Spaltenobjekte überprüfen. Bei Spaltenobjekten, die keine Defaultwerte besitzen, liefert der Agent den Wert `NoSuchInstance` zurück. Bei Objekten, die der Agent überhaupt nicht unterstützt, wird der Wert `NoSuchObject` an den Manager gesendet. Mit Hilfe dieser Informationen kann der Manager nun die noch fehlenden Werte in einem abschließenden `set` Befehl setzen und den Status auf `active` umschalten.

#### 7.2.3.2.2 CreateAndGo

Im Gegensatz dazu erlaubt es die `CreateAndGo` Methode, eine Tabellenzeile in einem einzigen `set` Befehl zu erzeugen und zu aktivieren. Dazu setzt der Manager alle `read-create` Spaltenobjekte auf ihre entsprechenden Werte und das Statusobjekt auf den Wert `CreateAndGo`. Der Agent erzeugt daraufhin eine neue Tabellenzeile, initialisiert die Spaltenobjekte entsprechend und setzt das Statusobjekt auf `active`.

### 7.2.3.3 Löschen von Tabellenzeilen

Um eine Tabellenzeile zu löschen setzt der Manager den Wert des Statusobjektes auf `destroy`. Der Agent entfernt daraufhin die entsprechende Zeile. Der Manager hat auch die Möglichkeit eine Tabellenzeile vorübergehend „auszuschalten“, indem er das Statusobjekt auf den Wert `NotInService` setzt.

Die folgende Abbildung stellt alle möglichen Zustände einer Tabellenzeile, sowie die erlaubten Übergänge zwischen den Zuständen dar.

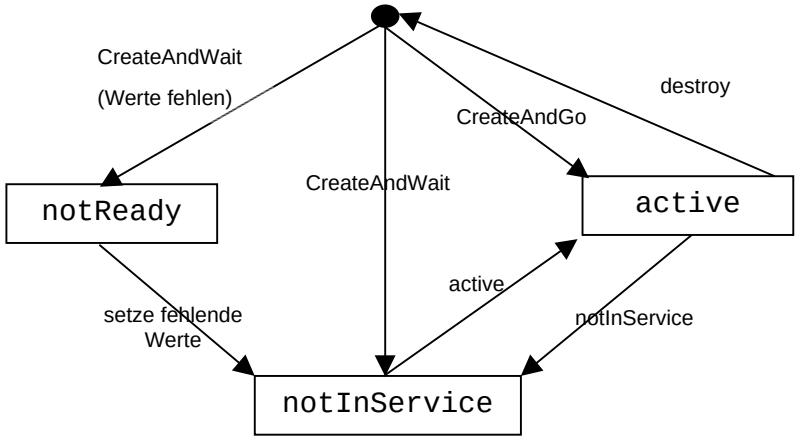



*Bild 7-1 Zustände einer Tabellenzeile*

Tritt bei der Bearbeitung von Tabellenzeilen ein Fehler auf (z.B. der Manager stellt einen falschen Wert zur Verfügung), so geht die Tabelle in den Zustand `notReady` und der Manager muss den Fehler erst beheben, bevor er die Tabellenzeile auf `active` setzen kann.

### 7.2.3.4 Definition von Traps

Für die Beschreibung von SNMPv2 Traps wurde ein neues `Makro NOTIFICATION-TYPE` definiert:

```
NOTIFICATION-TYPE   MACRO  ::=  BEGIN

   TYPE  NOTATION   ::= “OBJECTS“       { Objects} | empty
         “STATUS“         Status
         “DESCRIPTION“          Text
         „REFERENCE“            Text | empty
   VALUE NOTATION  ::= value (VALUE NotificationName)
END
```

Das Makro lässt sich am besten an einem einfachen Beispiel erklären. In der erweiterten Interface Gruppe der MIB-II wurde der `linkup` Trap folgendermaßen definiert:

```
linkup  NOTIFICATION-TYPE
     OBJECTS       {ifIndex, ifAdminStatus, ifOperStatus }
     STATUS        current
     DESCRIPTION  „A linkUp trap signifies that the SNMPv2
entity, acting in an agent role, has detected that the
ifOperStatus object for one of ist communication linkes has
transitioned out of the down state.“
::= { snmpTraps 4 }
```

Diese Trap Beschreibung besagt somit, dass ein `linkUp` Trap immer dann von einem Agenten gesendet wird, falls eine Kommunikationsverbindung (wieder) hergestellt werden konnte. Das `OBJECTS` Feld besagt dabei, dass der Agent die Objekte `ifIndex`, `ifAdminStatus` und `ifOperStatus` (und zwar genau in dieser Reihenfolge) in dem `VarBindList` Feld des Traps mitsendet.

Für jeden SNMPv2 Trap wird zusätzlich zu den optionalen *Managed* immer der *Object Identifier* des Traps (hier `snmpTraps 4`) sowie die aktuelle Zeit (`sysUpTime`) des Traps mitgeliefert.

## 7.3 Erweiterungen der MIB-II

Die Erweiterungen in der SNMPv2 *Management Information Base* beziehen sich im wesentlichen auf die folgenden Gruppen:

- `system` Gruppe — enthält Erweiterungen zu der MIB-II `system` Gruppe
- `SNMP` Gruppe — enthält Objekte für das SNMP Protokoll selbst
- `interfaces` Gruppe — enthält Objekte zur Beschreibung der Schnittstellen
- `MIB objects` Gruppe — enthält Objekte für Trap PDUs sowie für die Koordination von `set` Operationen

Die Änderungen in den ersten drei Gruppen resultieren aus dem praktischen Einsatz von SNMP. Im wesentlichen wurden neue nützliche Objekte definiert oder wie bei der SNMP Gruppe viele nicht sehr sinnvolle Objekte gestrichen. Für eine detailliertere Beschreibung der Gruppen in der SNMPv2 MIB sei auf RFC1907 verwiesen. Im folgenden soll nur die MIB *Objects* Gruppe als einzige neue Gruppe in SNMPv2 vorgestellt werden.

### 7.3.1 MIB Objects Gruppe

Die `Objects` Gruppe enthält zwei Untergruppen, wie in der folgenden Abbildung dargestellt:

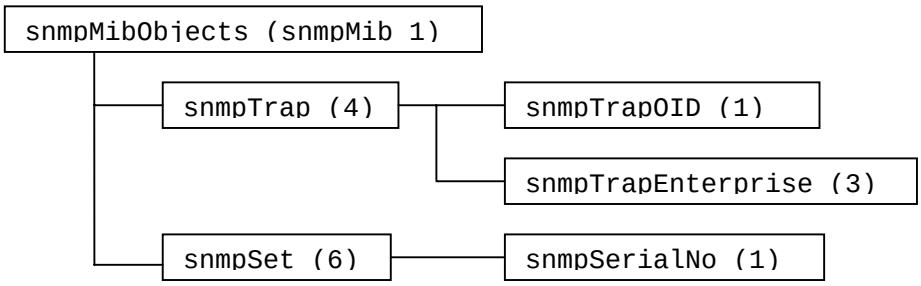



*Bild 7-2 SNMP MIB* Objects *Gruppe*

Die erste Untergruppe enthält zwei Objekte `snmpTrapOID` und `snmpTrapEnterprise`, die beim Senden von Traps oder `InformRequests` eine Rolle spielen:

- `snmpTrapOID` repräsentiert den *Object Identifier* des Traps oder der `inform`-Operation. Diese Variable erscheint immer an zweiter Stelle in jeder SNMPv2 Trap und `inform` PDU.
- `snmpTrapEnterprise` repräsentiert den „enterprise“ *Object Identifier* für den jeweiligen Trap.

Im zweiten Teil der MIB *Objects* Gruppe wurde eine zweite Untergruppe `snmpSet` definiert, die zur Zeit nur ein einzelnes Objekt `snmpSerialNo` vom Typ `INTEGER` enthält. Dieses Objekt wurde eingeführt, um Inkonsistenzen durch konkurrierende Managerzugriffe zu vermeiden. Dieses Objekt ist vom Typ `TestAndIncr` für den die folgende Konvention definiert wurde:

Der aktuelle Wert von `snmpSerial` sei **K**, dann gilt:

1. Falls der Agent eine `set` Operation mit dem Wert **K** für dieses Objekt empfängt, so inkrementiert der Agent `snmpSerialNo` und die `set`-Operation liefert den Wert **K** zurück.
2. Falls der Agent eine `set` Operation mit einem Wert ungleich **K** empfängt, so wird der Fehler `inconsistentValue` zurückgeliefert.

Möchte nun ein Manager A eine `set` Operation ausführen, so holt er sich zunächst den aktuellen Wert von `snmpSerialNo` und führt dann die gewünschte `set` Operation aus, wobei er zusätzlich versucht, das `snmpSerialNo` Objekt auf den zuvor abgefragten Wert zu setzen. Sollte in der Zwischenzeit ein anderer Manager B bereits eine entsprechende `set` Operation erfolgreich ausgeführt haben, so schlägt die set Operation des Managers fehl, da der Wert von `snmpSerialNo` durch die erstere `set` Operation bereits inkrementiert wurde.

Auf diese Weise kann garantiert werden, dass zu jeder Zeit immer nur ein Manager eine `set` Operation ausführen kann. Dies setzt natürlich voraus, dass jeder Manager

sich an diese Konvention hält. Es ist ebenfalls denkbar und auch sinnvoll, dass ein entsprechendes Objekt für jede Gruppe einer MIB definiert wird, um parallelen Zugriff von mehreren Managern auf diese Gruppen zu erlauben.

## 7.4 Erweiterungen des SNMP Protokolls

Die Erweiterungen des SNMP Protokoll gegenüber der Version SNMPv1 beziehen sich hauptsächlich auf die Einführung neuer Operationen sowie einzelne Verbesserungen im Bereich Effizienz und Fehlerbehandlung. Wie bei SNMPv1 teilt sich eine SNMPv2 Nachricht in drei Anteile: Versionsnummer, *Community*-Name und Datenteil.

Die Versionsnummer einer SNMPv2 Nachricht besitzt den **Wert 1** (0 für SNMPv1). Der Datenteil enthält die Protokolldateneinheiten der einzelnen Operationen.

In SNMPv2 wurden neben den bereits bestehenden `get`, `get-next` und `set` Operationen zwei zusätzliche Operationen definiert:

- `get-bulk` .. für das effiziente Auslesen von Tabellen
- `inform` .. für die Kommunikation zwischen Managern

Des weiteren wurde die Struktur der SNMPv1-Trap-PDU an die PDUs der übrigen Operationen angepasst. Da auch die beiden hinzugekommenen Operationen denselben PDU Aufbau besitzen, gibt es für SNMPv2 nunmehr nur noch eine PDU, in der alle Operationen verpackt werden können.

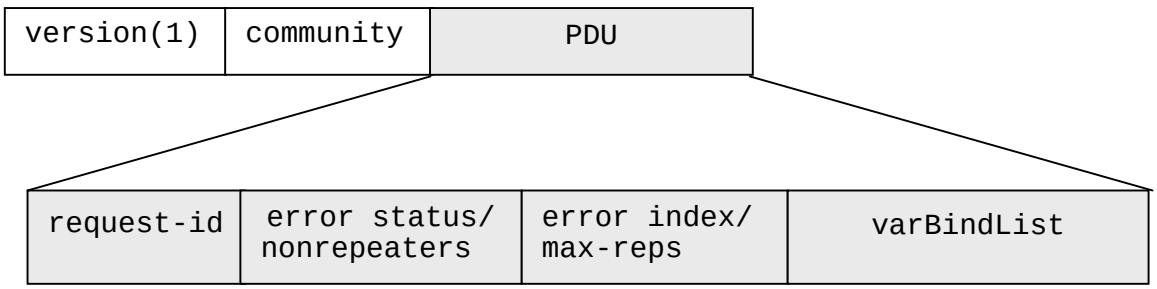



*Bild 7-3 Aufbau einer SNMPv2 Nachricht*

Allerdings besitzt die `get-bulk` Operation anstelle des `error-status/index` Feldes zwei Felder mit einer für die `get-bulk` Operation spezifischen Bedeutung. Im folgenden sollen nochmals alle SNMPv2 Operationen vorgestellt werden:

### 7.4.1 Der get Operator

Die PDU der `get` Operation ist identisch zu der SNMPv1 PDU: Der einzige Unterschied besteht in der Art und Weise wie Ergebnisse zurückgegeben werden. Eine SNMPv1 `get` Operation ist atomar, d.h. entweder werden alle gewünschten *Managed Object*s zurückgegeben oder - falls ein Fehler auftreten sollte - überhaupt keine. Dies führt in der Praxis häufig dazu, dass SNMP `get` Operationen mehrfach gesendet werden müssen, falls ein Agent ein *Managed Object* nicht unterstützt.

In SNMPv2 kann ein Agent das Wertefeld eines *Managed Object* auf die folgenden zwei **Ausnahmebedingungen** (*exception conditions*) setzen, falls der Agent den Wert des Objekts nicht ermitteln kann:

- `noSuchObject` .. das Objekt existiert nicht in der MIB des Agenten
- `noSuchInstance` .. die Instanz des Objekts existiert nicht. Diese Fehlermeldung wird immer dann vom Agenten zurückgegeben, wenn ein Manager versucht auf ein Objekt in einer Tabelle mit einem falschen Index zuzugreifen.

Damit ist es für den Agenten möglich, eine `get`-Operation nur teilweise zu beantworten. Auf der anderen Seite erkennt der Manager sofort, wieso der Agent nicht die gewünschten Informationen liefern kann.

### 7.4.2 Der get-next Operator

Die `get-next` PDU ist identisch zu der entsprechenden SNMPv1 PDU. Die Semantik einer SNMPv2 `get-next` Operation wurde auch hier um eine Ausnahmebedingung `endOfMibView` erweitert. Diese Ausnahmebedingung wird dann vom Agenten gesetzt, falls eine `get-next` Operation auf die am Ende stehende Variable einer MIB ausgeführt werden soll.

### 7.4.3 Der set Operator

Die SNMPv2 `set` PDU ist ebenfalls identisch zu der PDU der Vorgängerversion. Allerdings wurden die Fehlercodes für die `set` Operation erheblich verfeinert. Gegenüber 5 möglichen Fehlercodes in SNMPv1 definiert SNMPv2 18 verschiedene Fehlerarten, die es dem Manager ermöglichen die genaue Ursache für das mögliche Scheitern einer `set` Operation herauszufinden.

### 7.4.4 Der get-bulk Operator

Ein großes Problem bei SNMPv1 ist der relativ hohe Managementverkehr, der das Netz zusätzlich belastet. Gerade für den Bereich des Fehlermanagements arbeitet SNMP deswegen sehr ineffizient, die Relation zwischen der Anzahl der abgefragten Daten und den dadurch erkannten Problemsituationen fällt sehr schlecht aus.

Um die Anzahl der SNMP-Pakete zu vermindern, können mit dem neuen `get-bulk`-Kommando in einem einzigen Paket ganze Tabellen übertragen werden. SNMPv1 Systeme wiederholen, wenn sie mehrere Variablen eines Agenten abfragen wollen, für jeden Parameter jeweils das `get-next`-Kommando. Um beispielsweise die Routing-Tabelle eines Routers (eine unter Umständen sehr umfangreiche Tabelle) abzufragen, muss eine SNMP-Managementstation für jeden Eintrag ein `get-next`-Kommando absetzen.

Die `get-bulk` Operation verwendet dasselbe Auswahlprinzip wie die `get-next` Operation, d.h. der Agent liefert immer den in der MIB-Hierachie folgenden Wert zurück. Im Unterschied zur `get-next` Operation können in der `get-bulk` PDU zugleich mehr als nur eine Nachfolgervariable zurückgegeben werden.

Die get-bulk PDU definiert dazu zwei Felder non-repeaters und max-repetitions, die in keiner anderen PDU auftauchen. Das erste Feld enthält die Anzahl der MIB-Variablen im varBindList Feld, für die jeweils nur ein Nachfolger zurückgeliefert werden soll. Für die restlichen Variablen im varBindList Feld zeigt max-repetition an, wie viele Nachfolgevariablen zurückgegeben werden sollen.

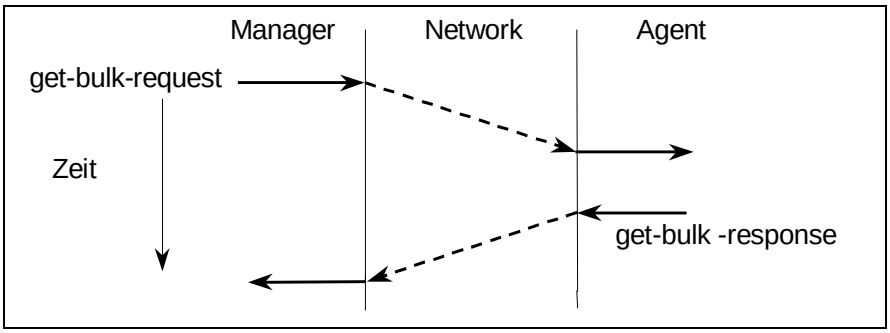



*Bild 7-4 get-bulk Operator*

Mit dem folgenden Befehl könnte man zum einen die Zeit seit dem letzten Reboot und zum anderen eine textuelle Beschreibung, den Typ und die Geschwindigkeit der ersten 5 Einträge in der MIB-II Interface Tabelle auslesen:

```
get-bulk (non-repeaters=1, max-repetitions=5, sysUpTime,
          ifDescr ifType, ifSpeed)
```

Der Agent kann aus den folgenden Gründen die Bearbeitung einer get-bulk Operation abbrechen:

- die Größe der PDU überschreitet ein (lokales) Maximum
- das Ende der MIB wurde erreicht (in diesem Fall enthalten die letzten Variablen im VarBindList Feld den Wert endOfMibView)
- die Verarbeitung der gesamten Operation ist für den Agenten zu zeitaufwendig

In jedem Fall liefert der Agent aber immer so viel Informationen zurück wie er kann und bricht nicht einfach die Operation komplett ab. Damit kann der Manager die noch fehlenden Informationen in einer zweiten get-bulk Operation abfragen.

### 7.4.5 Der inform Operator

Eine der wichtigsten Neuerungen von SNMPv2 ist die **Manager-zu-Manager Kommunikation**, dies bedeutet, dass eine Station als Manager oder/und als Agent agieren kann. Damit können **hierarchische Management-Strukturen** aufgebaut werden, in denen untergeordnete Manager Aufgaben für eine zentrale Managementstation übernehmen können. In der folgenden Abbildung Bild 7-5 ist ein solches, zweistufiges Konzept dargestellt, in dem

- die **lokalen Manager** die Komponenten der lokalen Netze überwachen und die darüberliegenden Ebenen nur bei wichtigen Ereignissen informieren,
- und das **Network Management Center (NMC)** primär die Verbindungen zwischen den LANs überwacht und nur die globale Sicht der Lokalen Netze hat.

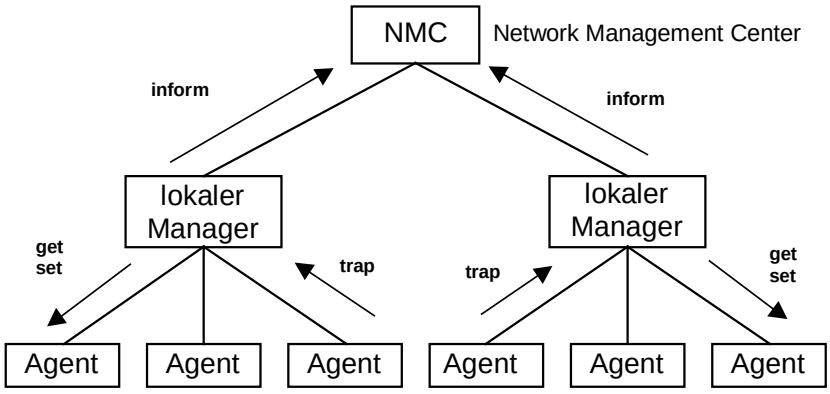



*Bild 7-5 Hierarchisches Management*

Weil diese Manager somit die zentrale Station entlasten, lassen sich weit größere SNMP-Installationen als bisher realisieren.

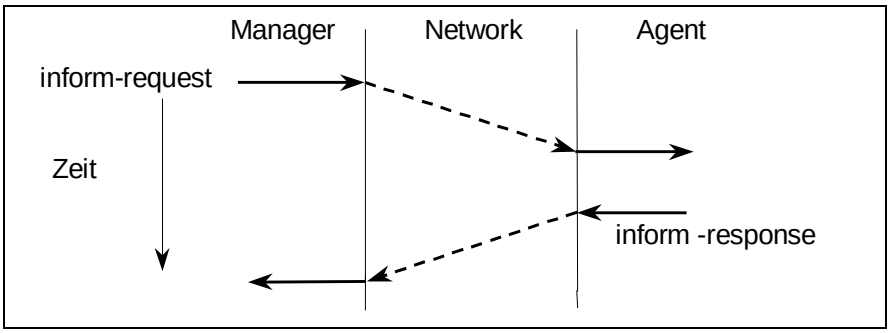



*Bild 7-6 `inform` Operator*

Die `inform` PDU ist genauso aufgebaut wie auch die `get/get-next` oder `set`-PDU. Die ersten beiden Variablen innerhalb des `VarBindList` Feldes sind immer eine Zeitmarke (`sysUpTime`) und ein Ereignis-Identifier. Ein Unterschied zu Traps besteht darin, dass `inform` Operationen vom Empfänger bestätigt werden müssen.

Obwohl ursprünglich für die Manager-zu-Manager Kommunikation gedacht, wird der `inform`-Operator häufig auch zwischen Agent und Manager als eine Art bestätigter Trap eingesetzt.

### 7.4.6 Der SNMPv2 trap Operator

Die SNMPv2 `trap` Operation erfüllt denselben Zweck wie die `trap` Operation von SNMPv1, allerdings mit einem unterschiedlichen PDU Format. SNMPv2 Traps verwenden dasselbe PDU Format wie alle anderen Operationen auch, was letztendlich die Implementierung von SNMP vereinfacht.

Das `varBindList` Feld einer SNMPv2 `trap` Operation enthält neben möglichen spezifischen Variablen immer die folgenden Objekt Werte-Paare:

- `sysUpTime.0` aktuelle Zeit beim Auftreten des Traps

- `smpTrapOID.0` diese MIB-Variable ist Teil der erweiterten SNMPv2 MIB und enthält als Wert den *Object Identifier* des jeweiligen Traps.

SNMPv2 Traps werden mit Hilfe des `NOTIFICATION-TYPE` Makros definiert.

## 7.5 SNMPv2 Versionen

Die größten Veränderungen fanden im Bereich der Sicherheit statt, da dort die größten Defizite bestanden. In SNMPv1 werden die Kommunikationsbeziehungen zwischen Managementstationen und Agenten durch deren Zugehörigkeit zu Gruppen (*Communities*) definiert. Dieses Modell wurde in SNMPv2 durch die Einführung der "**SNMPv2-Party**" und des "**SNMPv2-context**" wesentlich erweitert.

Beide Konzepte konnten sich jedoch in der Internet-Gemeinde aufgrund ihrer Komplexität nicht durchsetzen und sind nur noch von historischem Interesse. Aus diesem Grund soll an dieser Stelle auch nicht näher auf die Sicherheitskonzepte von SNMPv2 eingegangen werden.

Die fehlende Akzeptanz von Seiten der Anwender führte letztendlich dazu, dass die SNMPv2 Sicherheitskonzepte nach einer Überarbeitung des SNMPv2 Standards (RFC1901-RFC1908) komplett fallengelassen wurden. Statt dessen wurde das von SNMPv1 bekannte *Community* Modell übernommen.

### 7.5.1 SNMPv2c

Diese als ***Community based*** SNMPv2 bezeichnete Version stellte den kleinsten gemeinsamen Nenner dar und hat den Vorteil, dass sie schnell und einfach in die Praxis umzusetzen ist. Jedoch ist die Verwendung des unsicheren, nicht mehr zeitgemäßen Sicherheitsmodells von SNMPv1 ein gravierender Nachteil. Aus diesem Grund gingen die Arbeiten weiter, die zu zwei konkurrierenden Ansätzen führten.

### 7.5.2 SNMPv2u

SNMPv2u ähnelt in vielen Bereichen der ursprünglichen Spezifikation von SNMPv2. Anstelle der *parties* tritt hier ein **User/Agent-Konzept**, daher der Name User-Based-Security Model (oder kurz USEC-Modell). Der Anwender authentisiert sich mittels Name und zugehörigem Passwort. Zu jedem Anwender kann ein Schlüssel generiert werden, der dann zur Verschlüsselung der übermittelten Daten benutzt wird.

### 7.5.3 SNMPv2*

Der zweite Ansatz zur Lösung des Sicherheitsproblems wird als SNMPv2* (*v2 star*) bezeichnet, das zu etwa 80% auf den Dokumenten zu SNMPv2u basiert. Zusätzlich zu den bei SNMPv2u beschriebenen Punkten beschäftigt sich SNMPv2* noch mit Funktionen, die auf der Managementstation implementiert werden und mit der Möglichkeit der Fernkonfiguration eines Agenten, um beispielsweise die MIB eines Agenten von der Managementstation aus zu verändern.

# 8 Der dritte Anlauf: SNMPv3

## 8.1 Einführung

Nach dem Scheitern der SNMPv2 Arbeitsgruppe wurde die SNMPv3 Arbeitsgruppe ins Leben gerufen, die mit Hochdruck an einem einheitlichen Standard als Nachfolger von SNMPv1/2 arbeitet. Die Ziele der SNMPv3 Arbeitsgruppe wurden gleich zu Beginn klar definiert:

- Es sollte soviel wie möglich von den zwei konkurrierenden Versionen SNMPv2u und SNMPv2* übernommen werden.
- Es sollte ein Sicherheitskonzept entwickelt werden, das es erlaubt die `set` Operation über das weltweite Internet zu verwenden.
- SNMPv3 sollte eine flexible Architektur bekommen, die
  - den Einsatz in einer Vielzahl von Umgebungen berücksichtigt (von kleinen kostengünstigen Umgebungen mit wenig Funktionalität bis hin zu großen weltweiten Netzen),
  - es ermöglicht, einzelne Teilbereiche der Architektur weiterzuentwickeln auch wenn die Entwicklung der Gesamtarchitektur noch nicht abgeschlossen ist,
  - und es ermöglicht, alternative Modelle zu integrieren.
- SNMPv3 sollte so einfach wie möglich gehalten werden.

Im Januar 1998 wurden die folgenden Dokumente der SNMPv3 Arbeitsgruppe als ***Proposed Internet Standard*** veröffentlicht:

RFC2271 *An Architecture for Describing SNMP Management Frameworks*

RFC2272 *Message Processing and Dispatching for the SNMP Protocol*

RFC2273 *SNMPv3 Applications*

RFC2274 *User-based Security Model (USM) for version 3 of the SNMP Protocol*

RFC2275 *View-based Access Control Model (VACM) for the SNMP Protocol*

Inzwischen hat die IESG den SNMPv3 Spezifikationen den Status eines ***Draft Internet Standard*** zuerkannt; erste Referenzimplementierungen sind bereits verfügbar. SNMPv3 ist dabei kein neuer Standard für das Netzmanagement. So wurden z.B. keine neuen PDUs definiert. Vielmehr wurde der funktionale Anteil von SNMPv2 komplett in SNMPv3 übernommen. Neu in SNMPv3 ist das Architekturmodell, sowie die Arbeiten zum Thema Sicherheit. Dies wird in dem Einführungsdokument der SNMPv3 Arbeitsgruppe folgendermaßen zum Ausdruck gebracht:

*„SNMPv3 is SNMPv2 plus security and administration“*

In den folgenden Abschnitten sollen die grundlegenden Konzepte von SNMPv3 etwas näher erläutert werden.

## 8.2 Das Architekturmodell

Eines der wichtigsten Ziele der SNMPv3 Arbeitsgruppe war die Entwicklung einer flexiblen Architektur, die es erlaubt zukünftige Erweiterungen, beziehungsweise bereits bestehende Differenzen zwischen den verschiedenen Version einfach zu integrieren. Die Gesamtarchitektur, wie sie in RFC2271 beschrieben ist, besteht aus einer Anzahl von verteilten, interagierenden SNMP Einheiten. Eine SNMP Einheit kann dabei die Rolle eines Agenten, die eines Managers oder eine Kombination aus beiden annehmen.

### 8.2.1 SNMP Einheit

Jede SNMP Einheit besteht aus mehreren **diskreten Subsystemen**, die einen bestimmten Dienst bereitstellen sowie aus mehreren **Applikationen**, die diese Dienste in Anspruch nehmen. Für jedes Subsystem wurde eine abstrakte Dienstschnittstelle definiert, die es anderen Subsystemen oder Applikationen erlaubt auf diese Dienste zuzugreifen. Die folgende Abbildung zeigt den allgemeinen Aufbau einer solchen SNMP Einheit:

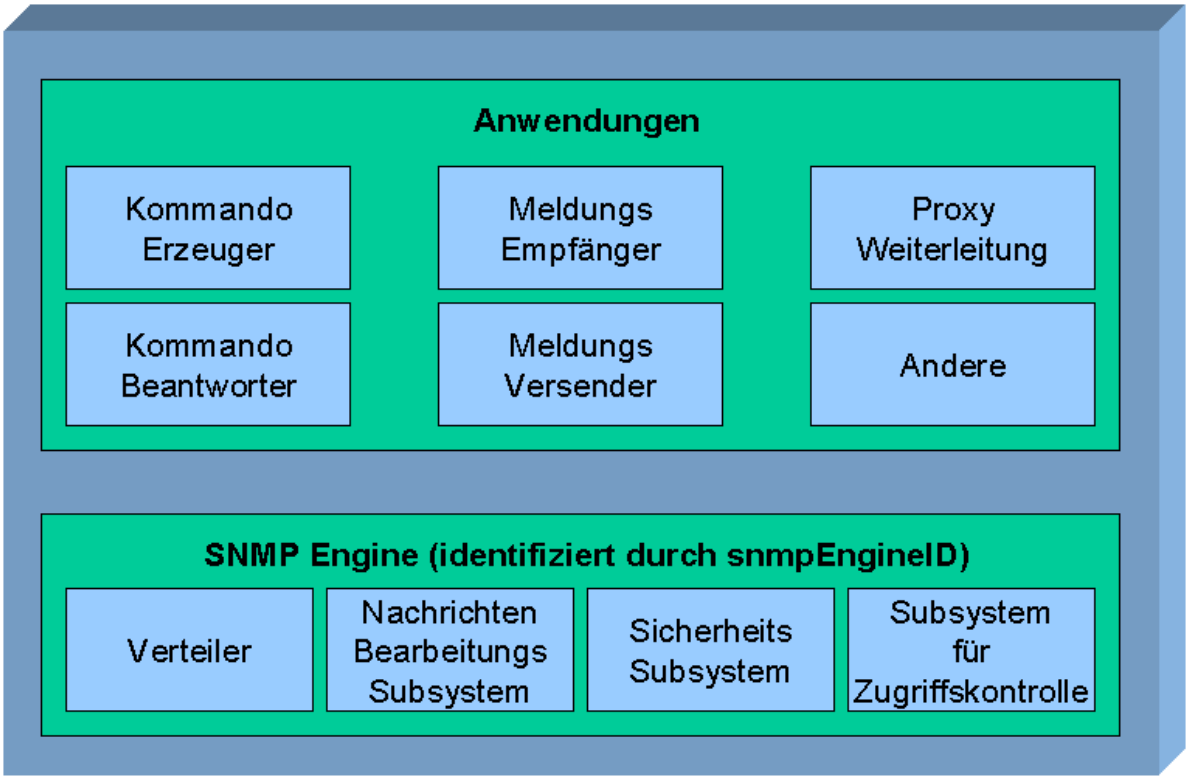



*Bild 8-1 Blockstruktur einer SNMP Einheit*

Die einzelnen Subsysteme einer SNMP Einheit werden zu einer SNMP Engine zusammengefasst. Eine SNMP Engine wird durch eine systemweit eindeutige Engine-ID identifiziert und stellt Dienste für das Senden und Empfangen von SNMP Nachrichten, für Authentifizierung und Verschlüsselung und für die Zugriffskontrolle zur Verfügung.

Die Applikationen sind ebenso wie die SNMP Engine in diskrete Module unterteilt und repräsentieren die Anwenderlogik in den Manager- und Agentenprozessen. Applikationen sind jedoch nicht Teil der SNMPv3 Spezifikation. Die folgenden Applikationen wurden bisher definiert:

- **Kommandoerzeuger** initiieren `get`, `get-next`, `get-bulk` und/oder `set` Operationen und verarbeiten die dazugehörigen Antworten.
- **Meldungsempfänger** behandeln asynchrone Meldungen (v1Trap-, v2Trap- und `inform` Operationen).
- **Kommandoempfänger** verarbeiten `get`, `get-next`, `get-bulk` oder `set` Operationen, überprüfen die Zugriffsrechte und erzeugen entsprechende Antworten.
- **Meldungserzeuger** generieren beim Auftreten bestimmter Ereignisse asynchrone Meldungen (v1Trap-, v2Trap- und `inform` Operationen).
- **Proxy Weiterleitung** leiten SNMP Nachrichten in transparenter Weise an andere SNMP-Einheiten weiter.

Je nach Zusammenstellung der einzelnen Subsysteme und Applikationen agiert die SNMP Einheit als Manager oder Agent, beziehungsweise als Kombination von beiden. Ein traditioneller SNMP Agent würde z.B. alle Subsysteme der SNMP Engine enthalten, sowie die Kommandoempfänger und Meldungsversender. Ein traditioneller SNMP Manager würde dagegen das Zugriffskontrollsubsystem nicht in seiner SNMP Engine enthalten dafür aber die Applikationen Kommandoerzeuger, Meldungsversender und Meldungsempfänger.

## 8.2.2 Abstrakte Dienstschnittstellen

Die Dienste, die von den einzelnen Subsystemen einer SNMP Einheit bereitgestellt werden, werden in den RFCs mit Hilfe von **Dienstelementen** (*service primitives*) definiert. Ein Dienstelement spezifiziert die Funktion die auszuführen ist und die Parameter, die verwendet werden, um Daten oder Kontrollparameter zu übergeben. Die tatsächliche Ausprägung eines Dienstelements wird nicht spezifiziert, sondern ist von der jeweiligen Implementierung abhängig (z.B. als Prozeduraufruf).

SNMPv3 unterscheidet drei Arten von Parametern: Eingabeparameter, Ausgabeparameter und Rückgabewert. Ein typischer Aufruf einer Dienstschnittstelle sieht folgendermaßen aus:

```
Rückgabewert = Funktionsanforderung(Eingabewert1, Eingabewert2, ...
                                    (Ausgabewert1, Ausgabewert2, ...)
```

Die Summe aller Dienstelemente eines Subsystems wird als **abstrakte Dienstschnittstelle** (*abstract service interface*) bezeichnet. Im folgenden Abschnitt werden die einzelnen Subsysteme näher beschrieben.

### 8.2.3 Der Verteiler (*Dispatcher*)

Der Verteiler ist die **zentrale Schaltstelle** innerhalb einer SNMP Einheit und stellt Dienste zum Versenden von PDUs und Empfangen von Nachrichten bereit. Applikationen (wie z.B. Kommandoerzeuger oder Meldungsversender) übergeben die SNMP PDUs, die sie versenden möchten, an den Verteiler. Der Verteiler wiederum sendet die PDU an das entsprechende Nachrichtenbearbeitungssubsystem. Das Nachrichtenbearbeitungssubsystem wandelt die SNMP PDU in ein entsprechendes SNMP Nachrichtenformat (SNMPv1, SNMPv2 oder SNMPv3) um und gibt die erzeugte SNMP Nachricht wieder an den Verteiler zurück, der die vollständige Nachricht auf das jeweilige Transportprotokoll überträgt und an den Empfänger sendet.

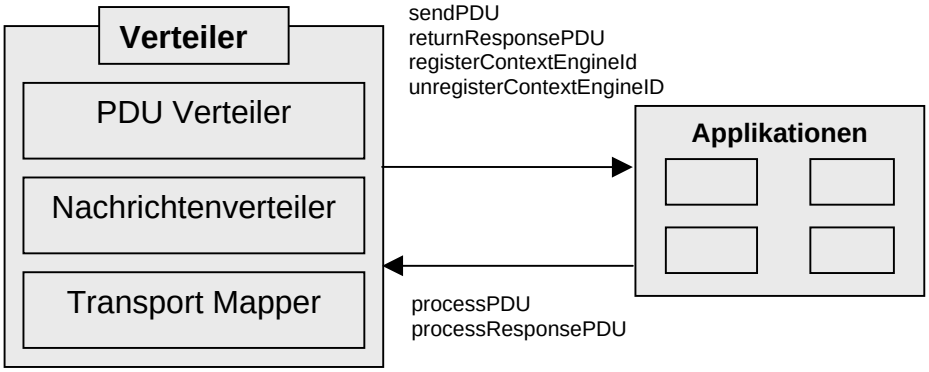



*Bild 8-2 Abstrakte Schnittstelle des Verteilers*

Im umgekehrten Fall empfängt der Verteiler SNMP-Nachrichten von der Transportschicht. Eingehende SNMP Nachrichten werden je nach Version an das entsprechende Nachrichtenbearbeitungssubsystem übergeben, das aus der Nachricht die SNMP PDU extrahiert und an den Verteiler zurückgibt. Die extrahierten PDUs werden nun vom Verteiler an die entsprechenden Applikationen verteilt.

Der Verteiler definiert die folgenden sechs Dienstelemente:

`sendPDU`: Applikationen, wie z.B. die Kommandoerzeuger Applikation, nutzen diese Funktion, um PDUs an andere SNMP Einheiten zu senden. Als Parameter werden die zu sendende PDU, das entsprechende Nachrichtenformat sowie verschiedene Sicherheitsparameter übergeben. Die Funktion liefert ein „Handle“ zurück, das der *request-ID* der SNMP Nachricht entspricht und für die Zuordnung der Antwort verwendet wird.

`processResponsePDU`: Diese Funktion wird von dem Verteiler benutzt, um eintreffende Antwort-PDUs den entsprechenden Applikationen zuzuordnen.

`processPDU`: Diese Funktion wird vom Verteiler benutzt, um eintreffende SNMP PDUs an die entsprechenden Applikationen zu verteilen.

`returnResponsePDU`: Applikationen, wie z.B. Kommandobeantworter Applikationen benutzen diese Funktion, um eine entsprechende PDU als Antwort einer vorhergehenden Anfrage oder Meldung zurückzusenden.

`registerContextEngineId`: Applikationen nutzen diese Funktion, um sich für bestimmte PDU Typen zu registrieren. Kommandoerzeuger Applikationen müssen sich beispielsweise für „response-PDUs“ anmelden.

`deregisterContextEngineId`: Applikationen nutzen diese Funktion, um sich von bestimmten PDU Typen abzumelden.

### 8.2.4 Nachrichtenbearbeitungssubsystem (*Message Processing Subsystem*)

Das **Nachrichtenbearbeitungssubsystem** ist für die Interpretation und Erzeugung von SNMP Nachrichten zuständig. Das Subsystem kann aus verschiedenen Nachrichtenbearbeitungsmodellen (*Message-Processing-Modellen***)** bestehen, wobei im aktuellen Ansatz Modelle für SNMPv1, SNMPv2C und SNMPv3 vorgesehen sind. Das für SNMPv3 entwickelte Nachrichtenformat ist in der folgenden Abbildung dargestellt:

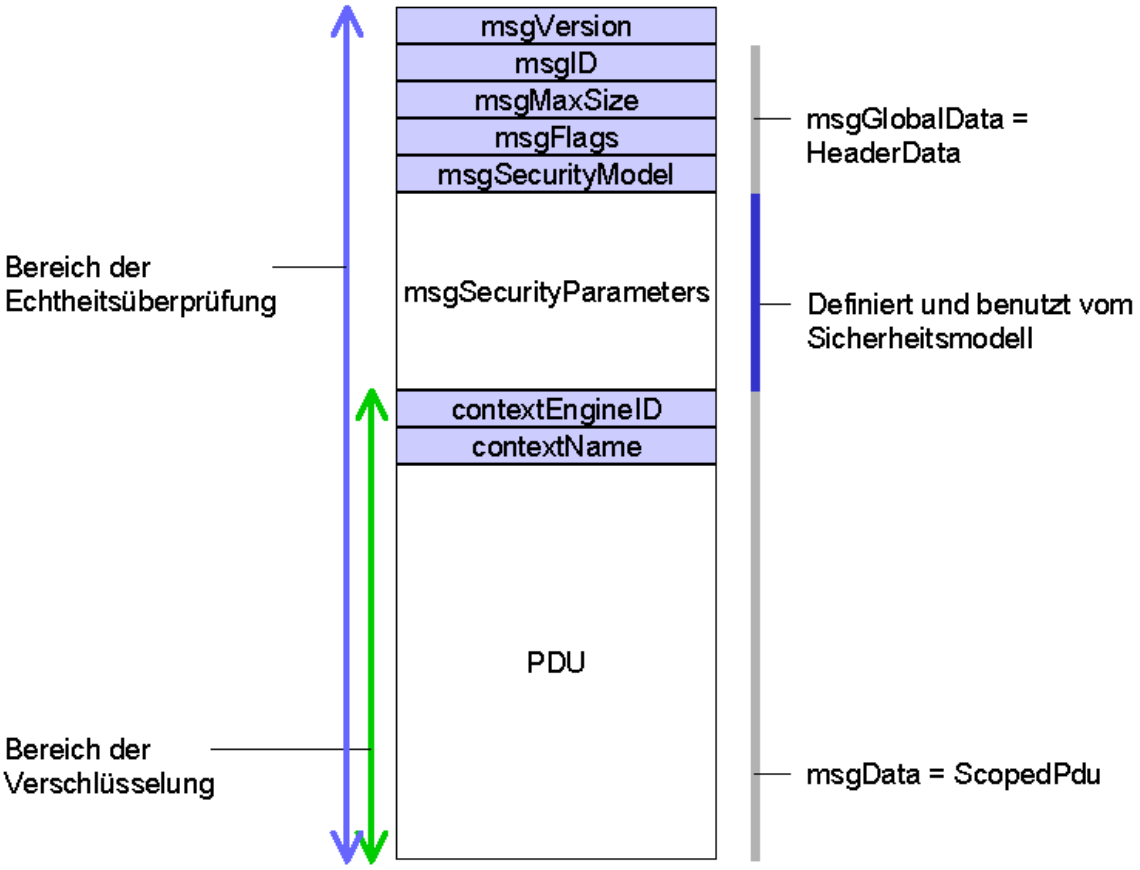



*Bild 8-3 Aufbau einer SNMPv3 Nachricht*

Erläuterungen der einzelnen Felder:

- `msgVersion`: Version der SNMP Nachricht (2 = SNMPv3)
- `msgID`: Eine eindeutige ID, welche von zwei SNMP Einheiten verwendet wird, um Anforderung und Antwort miteinander zu koordinieren.

- `msgMaxSize`: Beinhaltet die maximale Nachrichtengröße in *Octets*, welche der Sender zu verarbeiten vermag.
- `msgFlags`: Ein Octetstring, welcher in den 3 letzten Bits die Signale „`reportableFlag, privFlag, authFlag`“ beinhaltet. Sie haben folgende Bedeutung: Falls `reportableFlag` auf „1“ steht, so muss beim Empfang der Nachricht eine Antwort an den Absender erzeugt werden. Falls `privFlag` auf „1“ steht, so ist die Nachricht verschlüsselt. Falls `authFlag` auf „1“ steht, so wünscht sich der Absender auszuweisen. `privFlag` und `authFlag` ist in fast jeder Kombination möglich. Es ist jedoch nicht möglich, dass `privFlag` auf „1“ steht und `authFlag` auf „0“. Verschlüsselung ist nur mit Echtheitsüberprüfung erlaubt.
- `msgSecurityModel`: Eine Zahl zwischen 0 und $2^{31}$-1. Hiermit wird die Art des verwendeten Sicherheitsmodells angezeigt. Vordefiniert sind: 1 für SNMPv1, 2 für SNMPv2 und 3 für USM.
- `msgSecurityParameters`: Ein Octetstring, welcher vom Sender generiert und vom Empfänger interpretiert wird. Er enthält Parameter für das Sicherheitssubsystem.
- `contextEngineID`: Ein Octetstring, welcher die SNMP Einheit identifiziert. Bei eingehenden Nachrichten wird hiermit angezeigt, an welche Anwendung die PDU gesendet werden muss. Bei ausgehenden Nachrichten wird dieses Feld von der sendenden Anwendung gesetzt.
- `contextName`: Ein String, welcher den Kontext identifiziert.
- `data`: Die mitgesendete PDU. Bei SnMPv3 muss dieses eine SNMPv2 PDU sein.

Das Nachrichtenbearbeitungssystem berücksichtigt ausschließlich die ersten fünf Felder des Nachrichtenheaders, während die Sicherheitsparameter vom Sicherheitssubsystem interpretiert und gesetzt werden.

Die Schnittstellen des Nachrichtenbearbeitungssubsystems sind aus folgender Abbildung ersichtlich:

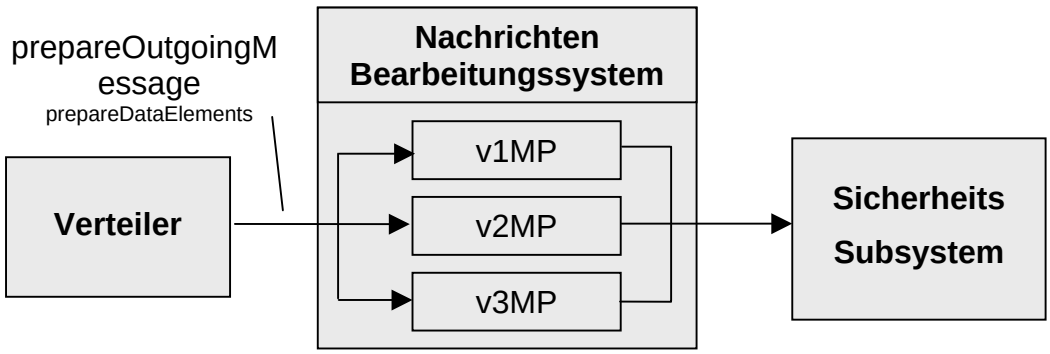



*Bild 8-4 Schnittstelle des Nachrichtenbearbeitungssystems*

Die Schnittstelle zum Nachrichtenbearbeitungssubsystem besteht aus den folgenden drei Dienstelementen:

`prepareOutgoingMessage`:
Diese Funktion wird vom Verteiler genutzt, um zu sendende PDUs in die entsprechende SNMP Nachricht zu konvertieren. Die Funktion erwartet eine PDU als Eingabeparameter und liefert die gewünschte SNMP Nachricht als Ausgabeparameter zurück.

`prepareDataElements`:
Diese Funktion wird vom Verteiler genutzt, um aus einer SNMP Nachricht die entsprechende PDU zu extrahieren.

`prepareResponseMessage`:
Diese Funktion wird vom Verteiler genutzt, um eine Antwort-PDU in eine entsprechende SNMP Nachricht umzuwandeln.

### 8.2.5 Sicherheitssubsystem (*Security Subsystem*)

Das Sicherheitssubsystem kann ebenfalls aus verschiedenen, parallel eingebundenen Sicherheits-Modellen (SM) bestehen. Das erste definierte SM in der SNMPv3 Architektur ist das ***User Based Security Model*** (USM), das zu einem Großteil von SNMPv2u übernommen wurde. Es muss die Datenintegrität und Authentizität sowie Vertraulichkeit und Aktualität der SNMP Nachrichten gewährleisten. Für verschiedene Sicherheitsanforderungen lassen sich Authentifizierung und Verschlüsselung wahlweise vorschreiben.

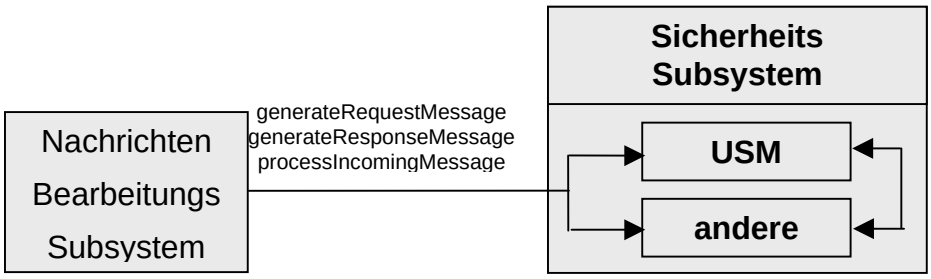



*Bild 8-5 Schnittstelle des Sicherheitssubsystems*

`generateRequestMessage`:
Diese Funktion führt je nach Sicherheitsstufe die entsprechende Authentifizierung und Verschlüsselung einer SNMP Nachricht durch und gibt das Ergebnis zusammen mit den entsprechenden Sicherheitsparametern zurück.

`generateResponseMessage`:
Diese Funktion entspricht der obigen Funktion mit dem Unterschied, dass diese Funktion für Antwort PDUs aufgerufen wird.

`processIncomingMessage`:
Diese Funktion erwartet eine eventuell authentifizierte und verschlüsselte SNMP Nachricht und überprüft die Authentizität bzw. führt eine Entschlüsselung der Nachricht durch.

## 8.2.6 Das Zugriffskontrollsubsystem

Das Zugriffkontrollsubsystem kontrolliert den Zugriff auf *Managed Objects* innerhalb eines Agenten und kann wie die anderen Subsysteme aus mehreren parallel eingebundenen Modellen bestehen. Für SNMPv3 ist bis jetzt nur das *View Access Control Model* (VACM) definiert.

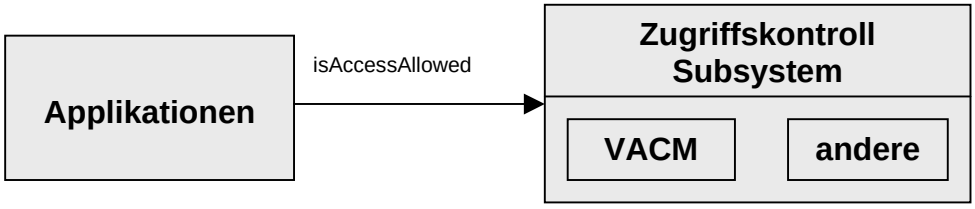



*Bild 8-6 Schnittstelle des Zugriffskontrollsubsystems*

Die Schnittstelle zum Zugriffskontrollsubsystem besteht aus nur einem Dienstelement:

`isAccessAllowed`: Diese Funktion wird von den Applikationen (z.B. Kommandobeantworter) aufgerufen, um den Zugriff auf *Managed Objects* des Agenten zu verifizieren. Die Funktion erwartet als Eingabeparameter den Name des Anwenders (*principal*), die Sicherheitsstufe (*security level*), die Art des Zugriffs, den Kontextnamen und den *Object Identifier* und gibt einen Wert zurück, der entweder den Zugriff gewährt (*access allowed*) oder aber verschiedene Fehlerursachen enthält.

## 8.2.7 Zusammenspiel der Applikationen und Subsysteme

Das Zusammenspiel zwischen den Applikationen und Subsystemen innerhalb einer SNMP Einheit wird in den folgenden Abschnitten exemplarisch dargestellt.

### 8.2.7.1 Initiieren einer SNMP Nachricht

In diesem Szenario wird beschrieben, wie eine Kommandoerzeuger Applikation eine SNMP PDU an eine andere SNMP Einheit sendet und die entsprechende Antwort wieder empfängt:

Um eine SNMP Nachricht an eine andere SNMP Einheit (z.B. einen Agenten) zu senden, ruft der Kommandoerzeuger die `sendPDU` Funktion des Verteilers auf und übergibt die zu sendende PDU sowie mehrere Parameter die das gewünschte Nachrichtenformat und Sicherheitsmodell definieren.

Der Verteiler leitet daraufhin die SNMP PDU an das entsprechende Bearbeitungssubsystem (für SNMPv1/v2/v3) weiter (`prepareOutgoingMessage`). Dieses generiert aus der PDU und den übergebenen Parametern eine entsprechende SNMPv1/2/3 Nachricht. Falls Authentifizierung oder Verschlüsselung erforderlich ist, wird die SNMP Nachricht an das Sicherheitssubsystem weitergeleitet (`generateRequestMsg`).

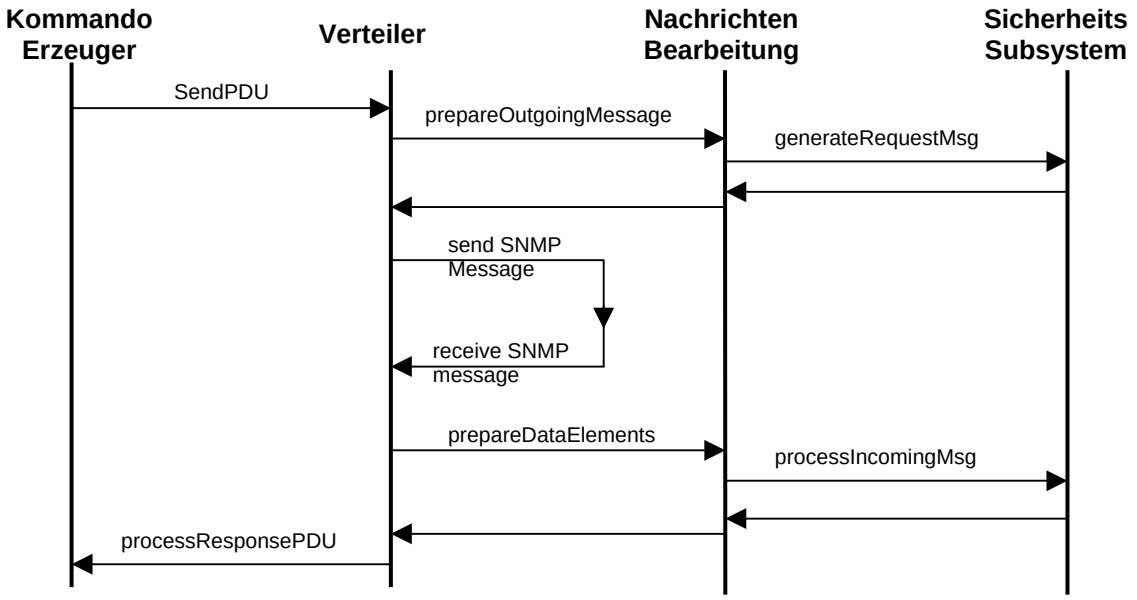



*Bild 8-7 Initiierung einer SNMP Nachricht*

Das Sicherheitssubsystem führt je nach Sicherheitsstufe und –modell eine Authentifizierung und/oder Verschlüsselung durch, setzt die entsprechenden Sicherheitsparameter in der SNMP Nachricht und gibt diese wieder zurück.

Der Verteiler überträgt nun die so generierte SNMP Nachricht auf das jeweilige Transportprotokoll und empfängt im Gegenzug die entsprechende Antwortnachricht, die er dann wiederum an das Nachrichtenbearbeitungssubsystem übergibt (`prepareDataElements`).

Das Nachrichtenbearbeitungssubsystem überprüft zunächst, ob die Nachricht authentifiziert oder verschlüsselt wurde. Falls ja, sendet sie die Nachricht an das Sicherheitssubsystem (`processIncomingMsg`), das die Nachricht entschlüsselt und auf Authentizität überprüft. Anschließend extrahiert das Nachrichtenbearbeitungs-subsystem die in der Nachricht enthaltene SNMP PDU und sendet diese an den Verteiler zurück.

Der Verteiler sendet nun die so gewonnene SNMP PDU an die Kommandoerzeuger Applikation zurück (`processResponsePDU`), der die Antwort PDU entsprechend bearbeitet.

### 8.2.7.2 Empfang einer SNMP Nachricht

Die folgende Abbildung zeigt das Szenario für das Empfangen und Verarbeiten einer SNMP Nachricht in einer SNMP Einheit (z.B. eines Agenten).

Bevor ein Kommandoempfänger SNMP PDUs vom Verteiler zugewiesen bekommt, muss er sich zunächst beim Verteiler dafür mit seiner SNMP *EngineID* anmelden (`registerContextEngineID`). Empfängt der Verteiler eine SNMP Nachricht, so wird daraus mit Hilfe der anderen Subsysteme die SNMP PDU extrahiert (siehe oben).

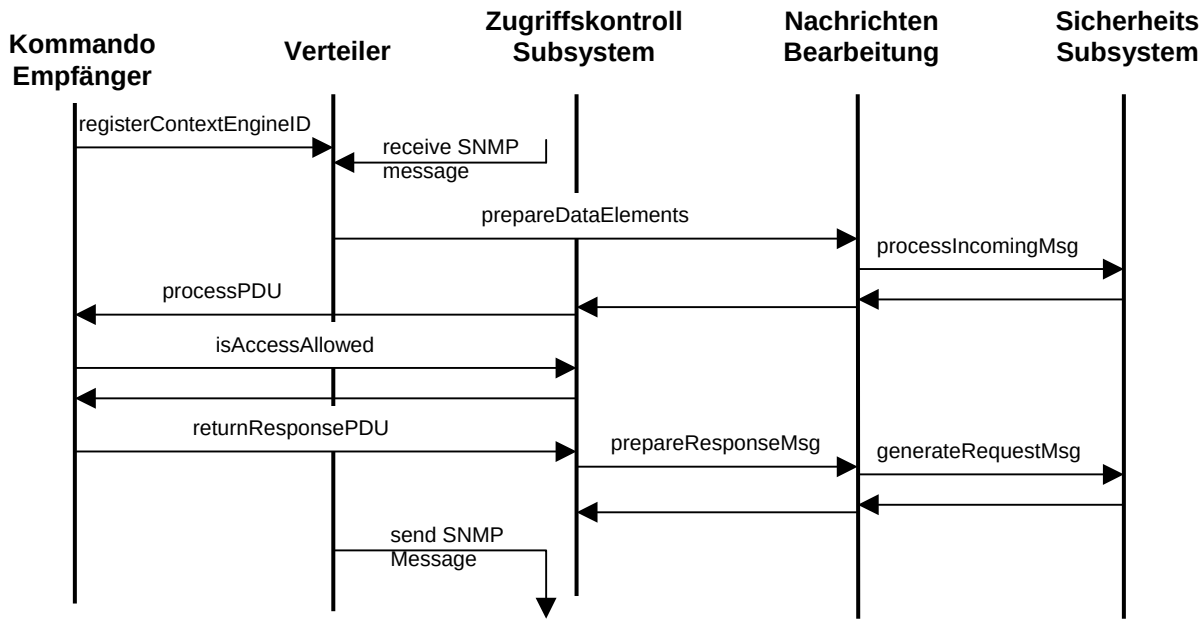



*Bild 8-8 Empfang und Verarbeitung einer SNMP Nachricht*

Der Verteiler ruft daraufhin die Funktion `processPDU` des Kommandoempfängers. Dieser überprüft zunächst, ob der Zugriff erlaubt ist, indem er die `isAccessAllowed` Funktion des Zugriffskontrollsubsystems aufruft. Ist dies der Fall so bearbeitet der Kommandoempfänger die PDU entsprechend, generiert eine Antwort PDU und sendet diese an den Verteiler zurück (`returnResponsePDU`).

Die Antwort PDU wird dann vom Verteiler an das Nachrichtenbearbeitungssubsystem weitergesendet (`prepareResponseMsg`), das eventuell mit Hilfe des Sicherheitssubsystems eine SNMP Nachricht daraus generiert und an den Verteiler zurückgibt. Dieser überträgt nun die SNMP Nachricht auf das entsprechende Transportprotokoll.

## 8.3 Das benutzerbasierte Sicherheitsmodell (USM)

Das benutzerbasierte Sicherheitsmodell für SNMPv3 wurde in RFC2274 definiert und umfasst die folgenden Punkte:

- **Aktualität**: Das SNMP Protokoll basiert typischerweise auf einem verbindungslosen Netzwerkdienst. Dadurch können Nachrichten verzögert, umgeordnet oder mehrmals wiedergegeben werden.
- **Authentifizierung**: Mit der Authentifizierung einer Nachricht wird die Identität der sendenden Einheit sichergestellt. Weiterhin garantiert eine Authentifizierung, dass eine Nachricht nicht modifiziert wurde.
- **Verschlüsselung**: Mit einer zusätzlichen Verschlüsselung einer Nachricht wird sichergestellt, dass es unberechtigten Personen nicht möglich ist, die ursprüngliche Nachricht mitzulesen.
- **Schlüsselverwaltung**: Authentifizierung und Verschlüsselung setzen voraus, dass beide SNMP Einheiten dieselben Schlüssel kennen. Dies wiederum bedingt, das sichere Übertragen von geheimen Schlüsseln über das Netzwerk sowie das Speichern von Schlüsseln in den lokalen SNMP Einheiten.

In den folgenden Abschnitten werden die Mechanismen des USM Sicherheitsmodell für die Umsetzung der obigen Ziele beschrieben.

### 8.3.1 Das Konzept der maßgebenden SNMP Einheit

Bevor auf die eigentlichen Sicherheitskonzepte eingegangen wird, ist es erforderlich, das Konzept einer **maßgebenden** (*authoritative*) SNMP Einheit zu definieren. Wann immer zwei SNMP Einheiten Nachrichten austauschen, übernimmt eine von beiden SNMP Einheiten die Rolle der maßgebenden SNMP Einheit:

- Bei SNMP Nachrichten, die eine Antwort erfordern (wie z.B. `get`, `set` oder `inform` Operationen), ist der Empfänger die maßgebende SNMP Einheit.
- Bei SNMP Nachrichten, die keine Antworten erfordern (wie z.B. `trap` Operationen), ist der Sender der Nachricht die maßgebende SNMP Einheit.

Die Unterscheidung zwischen einer maßgebenden und nicht-maßgebenden SNMP Einheit spielt vor allem bei Aktualität einer SNNP Nachricht, sowie bei der Schlüsselverwaltung eine Rolle.

### 8.3.2 Die Sicherheitsparameter einer SNMPv3 Nachricht

Die Sicherheitsparameter einer SNMPv3 Nachricht werden ausschließlich vom jeweiligen Sicherheitssubsystem interpretiert, bzw. gesetzt. Die folgende Abbildung gibt eine Übersicht über sämtliche Sicherheitsparameter:

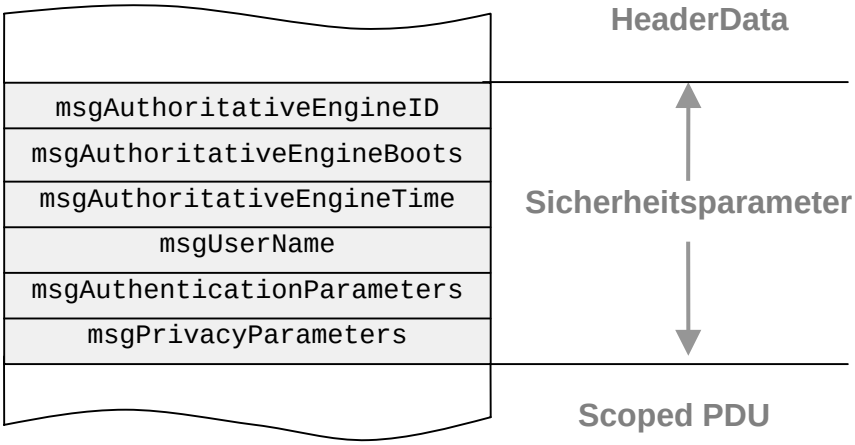



*Bild 8-9 Sicherheitsparameter einer SNMPv3 Nachricht*

Dabei besitzen die Parameter die folgenden Bedeutungen:

- `msgAuthoritativeEngineID`: SNMP-EngineID der maßgebenden SNMP Einheit.
- `msgAuthoritativeEngineBoots`: Anzahl der Bootvorgänge in der maßgebenden SNMP Einheit.
- `msgAuthoritativeEngineTime`: Zeit in Sekunden seit dem letzten Bootvorgang in der maßgebenden SNMP Einheit.
- `msgUserName`: Name des Anwenders, der die SNMP Nachricht initiiert.
- `msgAuthenticationParameters`: Ist leer, falls keine Authentifizierung vorgenommen wurde, andernfalls enthält dieser Parameter einen Authentifizierungscode.
- `msgPrivacyParameters`: Falls die Nachricht verschlüsselt wurde, enthält dieser Parameter einen für das Verschlüsselungsprotokoll spezifischen Anfangswert, ansonsten ist dieser Parameter leer.

### 8.3.3 Aktualität einer SNMP Nachricht

Die Aktualität einer SNMP Nachricht wird mit Hilfe zweier Zähler `snmpEngineBoots` und `snmpEngineTime` sichergestellt. Der erste Zähler gibt die Anzahl der Bootvorgänge an, während der zweite Zähler die Anzahl der Sekunden seit dem letzten Bootvorgang enthält. Eine SNMP Einheit im maßgebenden Modus muss beide Zähler verwalten und ständig aktualisieren.

Wenn zwei SNMP Einheiten miteinander kommunizieren, so muss sich die SNMP Einheit im nicht-maßgebenden Modus mit der anderen Einheit ständig

synchronisieren. Dazu verwaltet die SNMP Einheit im nicht-maßgebenden Modus für jede ihr bekannte *maßgebende* SNMP Einheit lokale Kopien der folgenden Variablen:

- `snmpEngineBoots`: der neueste geschätzte Wert des `snmpEngineBoots` Zählers der maßgebenden SNMP Einheit
- `snmpEngineTime`: der geschätzte Wert des `snmpEngineTime` Zählers der maßgebenden SNMP Einheit. Dieser Wert wird von der nicht-maßgebenden SNMP Einheit jede Sekunde inkrementiert.
- `latestReceivedEngineTime`: der letzte empfangene Wert des `snmpEngineTime` Zählers der maßgebenden SNMP Einheit.

**Synchronisierung**

Um eine lose Synchronisierung zwischen zwei SNMP Einheiten zu gewährleisten, sendet die SNMP Einheit im maßgebenden Modus die aktuellen Werte von `snmpEngineTime` und `snmpEngineBoots`, sowie ihre *SNMP* EngineID in jeder SNMP Nachricht mit. Falls die SNMP Nachricht authentisch ist und die folgende Bedingung erfüllt ist, aktualisiert die nicht-maßgebende SNMP Einheit ihre eigenen Kopien:

```
(msgAuthoritativeEngineBoots > snmpEngineBoots) OR
[(msgAuthoritativeEngineBoots = snmpEngineBoots) AND
(msgAuthoritativeEngineTime > latestReceivedEngineTime)]
```

Eine Synchronisierung zwischen zwei SNMP Einheiten findet nur dann statt, falls die SNMP Nachricht authentifiziert wurde. Diese Einschränkung ist erforderlich, da nur dann gewährleistet ist, dass die Werte der Zähler gültig sind.

**Zeitfenster**

SNMPv3 definiert ein Zeitfenster, in dem SNMP Nachrichten empfangen werden müssen. Dieses Zeitfenster hängt vom Modus der SNMP Einheit ab. SNMP Einheiten im maßgebenden Modus betrachten Nachrichten außerhalb dieses Zeitfensters, falls die folgende Bedingung zutrifft:

```
SnmpEngineBoots  =  2^31 – 1  OR
MsgAuthoritativeEngineBoots != snmpEngineBoots   OR
msgAuthoritativeEngineTime NOT IN [SnmpEngine-150,SnmpEngine+150]
```

Für SNMP Einheiten im nicht-maßgebenden Modus wird eine SNMP Nachricht zurückgewiesen, falls die folgende Bedingung zutrifft:

```
SnmpEngineBoots  =  2^31 – 1  OR
MsgAuthoritativeEngineBoots >= snmpEngineBoots   OR
[(msgAuthoritativeEngineBoots = snmpEngineBoots) AND
msgAuthoritativeEngineTime < snmpEngineTime –150)]
```

Damit ist die Überprüfung der Aktualität in einer nicht-maßgebenden SNMP Einheit toleranter.

### 8.3.4 Authentifizierung einer SNMP Nachricht

Die Authentifizierung einer SNMP Nachricht wird in SNMPv3 durch den Einsatz von zwei alternativen Protokollen erreicht: HMAC-MD5-96 und HMAC-SHA-96. HMAC ist der im Internet am weitesten verbreitete Algorithmus zur Authentifizierung von Nachrichten und erlaubt die Einbindung verschiedener **Hash-Funktionen**.

Eine Hash-Funktion erzeugt aus einer beliebig langen Nachricht einen sogenannten Hash-Code mit fester Größe als Ausgabe. Eine identische Nachricht erzeugt immer einen identischen Hash-Code, wobei es unmöglich ist, aus dem Hash-Code die Nachricht zurückzugewinnen. Das USM Sicherheitsmodell verwendet alternativ die Hash-Funktionen MD5 und SHA.

Der HMAC Algorithmus verbindet die Funktionsweise der Hash-Funktion mit der Benutzung eines privaten Schlüssels.

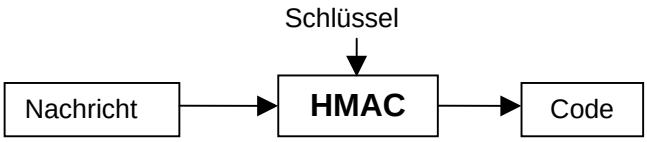



*Bild 8-10 Authentifizierung einer Nachricht*

Die Authentifizierung einer SNMP Nachricht erfolgt in drei Schritten:

1. Der Sender generiert aus der zu sendenden Nachricht mit Hilfe eines geheimen Schlüssels einen Code.
2. Dieser Code wird an die Nachricht angehängt und zusammen zum Empfänger geschickt.
3. Der Empfänger berechnet mit seinem Schlüssel und der empfangenen Nachricht denselben Code und vergleicht nun beide.

Wenn beide übereinstimmen, so lässt sich daraus schließen, dass die Nachricht nicht verändert wurde und dass die Nachricht vom richtigen Absender ist, da nur dieser den entsprechenden Schlüssel kennt.

Der vom Sender generierte Code wird in dem Sicherheitsparameter `msgAuthenticationParameters` einer SNMP Nachricht übertragen.

### 8.3.5 Verschlüsselung einer SNMP Nachricht

Mit der Verschlüsselung einer Nachricht wird sichergestellt, dass unberechtigte Personen den ursprünglichen Inhalt der Nachrichten nicht oder nur sehr schwer lesen können. In SNMPv3 wurde der DES-CBC Algorithmus (*Data Encryption Standard* ) verwendet. DES erlaubt die Verschlüsselung von 64 Bit Blöcken mit Hilfe eines 56 Bit Schlüssels. Um größere Datenmengen verschlüsseln zu können, müssen diese in 64 Bit Blöcke aufgeteilt werden.

Der in SNMPv3 festgeschriebene CBC (*Cipher Block Chaining*) Modus gewährleistet, dass wiederholende 64 Bit Blöcke zu einem unterschiedlichen verschlüsselten Text führen, um eine mögliche Entschlüsselung aufgrund von wiederkehrenden Mustern zu erschweren.

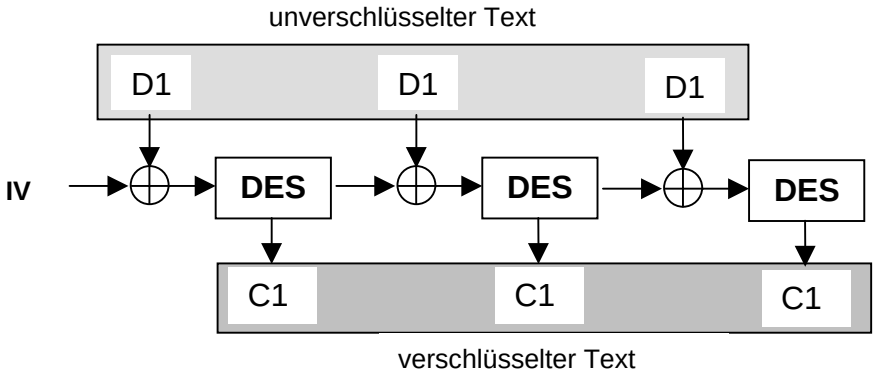



*Bild 8-11 CBC Modus des DES Verschlüsselungsalgorithmus*

Wie aus der obigen Abbildung ersichtlich benötigt der CBC Modus einen Anfangswert IV (*initial value*) für die Verschlüsselung. Dieser Anfangswert wird über einen sogenannten *salt value* berechnet, der im Sicherheitsparameter `msgPrivacyParameters` der SNMPv3 Nachricht übertragen wird. Die Verschlüsselung betrifft nur den *scoped PDU* Teil einer SNMP Nachricht.

## 8.3.6 Schlüsselverwaltung

Das Sicherheitsmodell in SNMPv3 erfordert, dass für jede Kommunikation zwischen einem Manager und einem Agenten ein privater Schlüssel für die Authentifizierung und ein privater Schlüssel für die Verschlüsselung von SNMP Nachrichten auf beiden Seiten bekannt sein muss. RFC2274 definiert Richtlinien für die Erzeugung, die Aktualisierung und die Verwaltung von Schlüsseln.

### 8.3.6.1 Erzeugung von Schlüsseln

Für Authentifizierung und Verschlüsselung muss der Anwender 16/20 Byte lange Schlüssel zur Verfügung stellen. Anwender arbeiten jedoch generell mit Passwörtern, die einfach gelesen und behalten werden können. In RFC2274 wurde deshalb der folgende Algorithmus definiert:

1. Man nehme das Passwort des Anwenders und wiederhole das Passwort so lange bis eine Zeichenkette von $2^{20}$ Bytes entsteht.
2. Wird ein 16 Byte Schlüssel benötigt, so nimmt man die MD5 Hash Funktion, wird ein 20 Byte langer Schlüssel benötigt, so nimmt man die SHA-1 Hash-Funktion.
3. Das Ergebnis der Hash-Funktion ist der Schlüssel für die Authentifizierung bzw. Verschlüsselung.

Ein weiterer Vorteil dieser Vorgehensweise ist, dass die Zeit zum „Knacken“ des Schlüssels erheblich zunimmt.

### 8.3.6.2 Verwaltung von Schlüsseln

Aus Gründen der Sicherheit, müsste jeder Anwender für jeden Agenten einen geheimen Schlüssel für die Authentifizierung und Verschlüsselung besitzen. Um die Anwender nicht mit einer Vielzahl von verschiedenen Schlüsseln zu belasten, erlaubt SNMPv3, dass jeder Anwender nur jeweils einen Schlüssel besitzen muss.

Gelingt es aber nun einem Angreifer, diesen einen Schlüssel herauszufinden, so ist er in der Lage, Managementzugriffe auf allen Agenten auszuführen. Um dies zu verhindern werden für jeden Agenten unterschiedliche Schlüssel verwendet, ohne jedoch den Anwender mit einer Vielzahl von unterschiedlichen Passwörtern zu belasten. Dies wird durch die folgende Prozedur erreicht:

1. Man nehme den vom Anwenderpasswort erzeugte Schlüssel und füge daran den Wert der Variablen `snmpEngineID` des Agenten an.
2. Falls ein 16 Byte langer Schlüssel benötigt wird, nimmt man die MD5 Hash Funktion, wird ein 20 Byte langer Schlüssel benötigt, so nimmt man die SHA-1 Hash Funktion.
3. Das Ergebnis ist ein lokaler Schlüssel, der für jeden Agenten unterschiedlich ist.

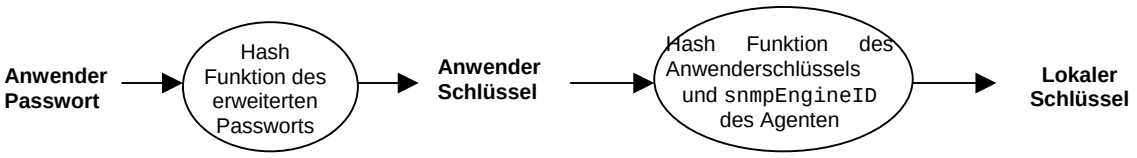



*Bild 8-12 Erzeugung von lokalen Schlüsseln*

Nur dieser lokale Schlüssel wird im Agenten gespeichert. Gelingt es einem Angreifer diesen lokalen Schlüssel aufzudecken, so kann er zwar damit auf diesen einen Agenten beliebig zugreifen, aber er hat keine Möglichkeit den eigentlichen Schlüssel aus dem lokalen Schlüssel zu generieren.

Möchte ein Manager nun auf einen Agenten zugreifen, so muss er zunächst den lokalen Schlüssel des Agenten nach der obigen Prozedur berechnen und kann dann mit diesem lokalen Schlüssel die Authentifizierung bzw. die Verschlüsselung der SNMP Nachricht vornehmen.

### 8.3.6.3 Aktualisierung von Schlüsseln

SNMPv3 spezifiziert nicht, wie Schlüssel erstmalig für einen Agenten eingerichtet werden. Dies ist Sache des Netzwerkadministrators und muss gegebenenfalls vor Ort geschehen. Aus Sicherheitsgründen sollten von Zeit zu Zeit diese Schlüssel geändert werden. Hierzu gibt es im allgemeinen zwei Möglichkeiten:

- Der neue Schlüssel wird mit dem alten Schlüssel verschlüsselt und an den Agenten gesendet.
- Der neue Schlüssel wird über einen nicht umkehrbaren Algorithmus verschlüsselt und an den Agenten gesendet.

Die erste Möglichkeit erscheint auf den ersten Blick am einfachsten, hat aber den Nachteil, dass der Agent die Verschlüsselung von Nachrichten unterstützen muss.

Dies ist aber bei einfachen Agenten nicht immer der Fall. In SNMPv3 wird deshalb der zweiter Ansatz verwendet, um sicher geheime Schlüssel über das Netz zu senden.

Um den neuen Schlüssel für unberechtigte Personen unleserlich zu machen wird der folgende Algorithmus angewendet:

1. Erzeuge eine beliebige Zufallszahl `random`
2. Berechne den Wert `digest:`

   ```
   digest = Hash ( keyOld || random )
   ```

   wobei Hash entweder die MD5 oder SHA-1 Hash-Funktion ist und das Symbol `||` eine Konkatenation darstellt.
3. Berechne daraus:

   ```
   protocolKeyChange = (random || (digest ⊕ keyNew))
   ```

   wobei ⊕ die XOR Operation ist.

Der Wert von `protocolKeyChange` wird dann in einem `set` Befehl über das Netzwerk gesendet. Um den neuen Schlüssel zu extrahieren muss der Agent die folgenden Schritte ausführen:

1. Berechne den Wert `digest`:

   ```
   digest = Hash ( keyOld || random )
   ```
2. Berechne daraus den neuen Schlüssel:

   ```
   keyNew = digest ⊕ delta
   ```

Nachdem der neue Schlüssel vom Agenten extrahiert wurde, trägt der Agent den Schlüssel in einer Tabelle (`usmUser` Gruppe) ein, womit die Aktualisierung des Schlüssels abgeschlossen ist.

## 8.4 Das sichtenbasierte Zugriffskontrollmodell (VACM)

SNMPv3 spezifiziert in RFC2574 das sichtenbasierte Zugriffskontrollmodell *(View-based access control model*). Die Zugriffskontrolle wird bei SNMPv3 immer dann in Anspruch genommen, wenn von außen auf ein Objekt lesend oder schreibend zugegriffen wird. Das Zugriffskontrollmodell stellt seine Dienste direkt den Applikationen Kommandobeantworter und Meldungserzeuger zur Verfügung.

### 8.4.1 Elemente des VACM Zugriffskontrollmodells

Das sichtenbasierte Zugriffskontrollmodell enthält die folgenden Elemente:

#### 8.4.1.1 Gruppen (*groups*)

Eine Gruppe ist definiert als ein Satz von `<securityModel, securityName>` Paaren. Der Parameter `securityModel` identifiziert das eingesetzte Sicherheitsmodell (z.B. USM) und `securityName` ist der Name Anwender

(*principal*), der die SNMP Nachricht initiiert hat. Eine Gruppe wird eindeutig durch einen Gruppennamen identifiziert. Alle Anwender einer Gruppe besitzen dieselben Rechte, wobei ein `<securityModel, securityName>` Paar immer nur Teil einer Gruppe sein kann.

#### 8.4.1.2 Sicherheitsstufe (*security level*)

SNMPv3 definiert drei verschiedene Sicherheitsstufen (`noAuthNoPriv`, `authNoPriv`, `authPriv`) für die unterschiedliche Zugriffsrechte definiert werden können. So ist es z.B. möglich, schreibende Zugriffe nur zuzulassen, falls die Nachricht authentifiziert und verschlüsselt wurde.

#### 8.4.1.3 Kontext (*context*)

Ein MIB Kontext identifiziert den Kontext, unter dem der Zugriff auf die *Managed Objects* einer MIB erfolgt. Ein MIB Kontext hat die folgenden Eigenschaften:

- Eine SNMP Einheit kann mehr als einen Kontext verwalten.
- Ein Objekt oder eine Objektinstanz kann in mehr als einem Kontext erscheinen.
- Falls eine SNMP Einheit mehr als ein Kontext besitzt, muss zusätzlich der `contextName` angegeben werden.

Als Beispiel für die Verwendung des Kontext Konzepts stelle man sich eine SNMP Einheit vor, die Managementinformationen für zwei Repeater bereitstellt. Dies könnte dann der Fall sein, wenn die Repeater selbst nicht SNMP fähig sind und so ein anderer Netzknoten im selben LAN als eine Art Proxy für die beiden Repeater arbeitet. Die SNMP Einheit auf diesem Netzknoten würde dann eine Repeater-MIB in zwei Kontexten „Repeater-1“ und „Repeater-2“ verwalten. Möchte ein Manager nun auf den ersten Repeater zugreifen, gibt er den Kontextnamen „Repeater-1“ an und für den zweiten Repeater „Repeater-2“. Damit weiß die SNMP Einheit, an welchen Repeater sie sich wenden muss, um die Anfrage des Managers zu beantworten.

#### 8.4.1.4 MIB-Sichten (MIB *views*)

Eine MIB-Sicht ist eine Untermenge von *Managed Objects* der lokalen MIB einer SNMP Einheit. Eine MIB-Sicht wird definiert als eine Sammlung von Teilbäumen, wobei ein Teilbaum entweder Teil der Sicht ist oder explizit von dieser Sicht ausgeschlossen ist.

Ein Teilbaum ist einfach definiert als ein Knoten in der MIB Hierarchie sowie alle darunterliegenden Knoten. Damit besitzen alle Elemente innerhalb eines Teilbaums einen gemeinsamen *Object Identifier* Präfix. Der Teilbaum wird durch diesen Präfix benannt und identifiziert.

### 8.4.2 Zugriffskontroll-Logik

Die abstrakte Dienstschnittstelle des Zugriffskontrollmoduls definiert nur eine Funktion `isAccessAllowed`. Diese Funktion erwartet insgesamt die sechs folgenden Variablen als Eingabeparameter für die Entscheidung:

- `securityModel`: das verwendete Sicherheitsmodell (z.B. *Community*-basiert oder USM)
- `securityName`: Name des Anwenders für den der Zugriff überprüft werden muss
- `securityLevel`: Sicherheitsstufe(`noAuthNoPriv`,`authNoPriv`,`authPriv`)
- `viewType`: bestimmt eine aus drei verschiedenen MIB-Sichten (`read`, `write` oder `notify`)
- `contextName`: Name des Kontextes unter dem der Zugriff gewährt werden soll
- `variableName`: *Object Identifier* und Instanz des Objektes

Die Vielzahl der Eingabeparameter erlaubt es dem Netzwerkadministrator den Zugriff auf die MIB eines Agenten sehr fein zu granulieren. Die zum Teil umfangreiche Konfiguration eines Agenten erfolgt in einer eigens dafür definierten VACM-*Management Information Base*. Diese MIB enthält mehrere Tabellen, die Informationen über die dem Agenten bekannten Kontexte, Gruppen und Views speichern. Der Netzwerkadministrator hat die Möglichkeit diese Tabellen über SNMP „remote“ zu konfigurieren.

Der Entscheidungsbaum für die Zugriffskontrolle ist in der folgenden Abbildung dargestellt:

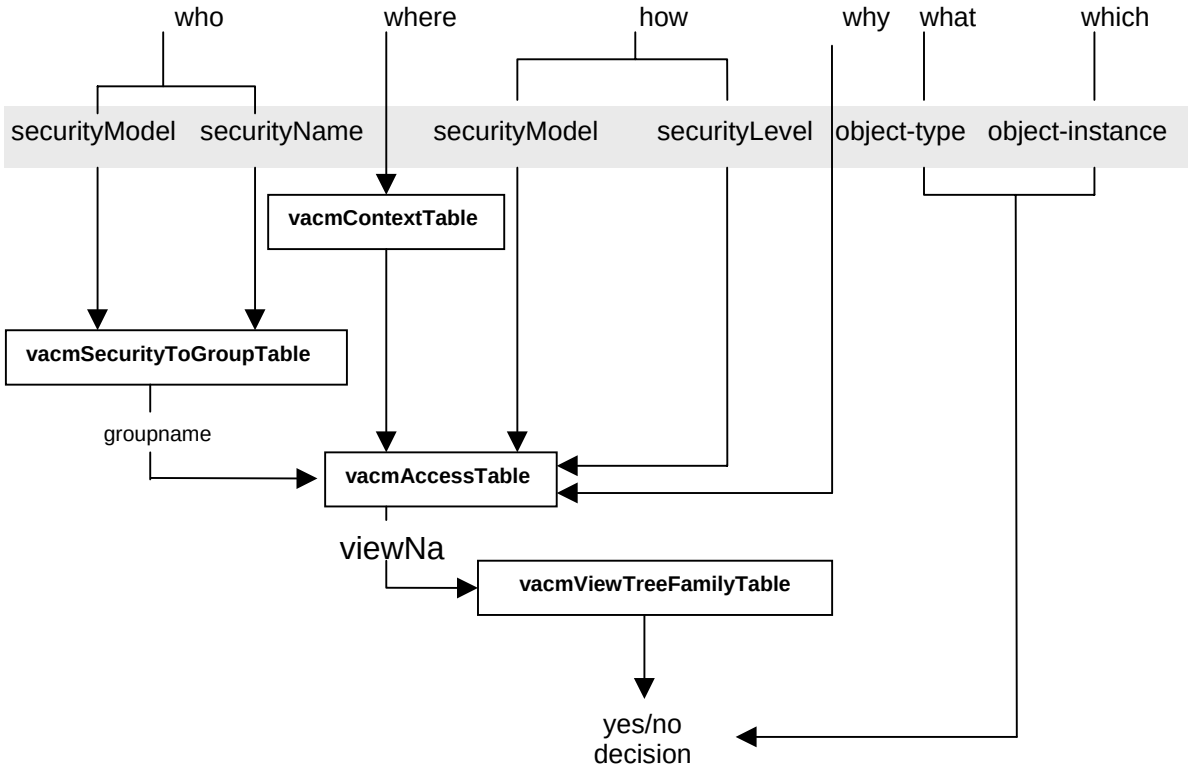



*Bild 8-13 Logik der VACM-Zugriffskontrolle*

Wenn die Funktion `isAccessAllowed` aufgerufen wird, führt die VACM Zugriffskontrolle die folgenden Schritte durch:

Die Zugriffskontrolle überprüft, ob der Kontext `contextName` in der `vacmContextTable` enthalten ist. Falls nicht wird der Fehler `noSuchContext` zurückgegeben und die Zugriffskontrolle abgebrochen.

Die Zugriffskontrolle überprüft, ob für die beiden Parameter `securityModel` und `securityName` ein Eintrag in der `vacmSecurityToGroupTable` Tabelle vorhanden ist. Falls ja, bedeutet dies, dass der Anwender unter dem angegebenen Sicherheitsmodell Teil einer Gruppe des Agenten ist. Falls nicht, wird der Fehler `noGroupName` zurückgegeben und die Zugriffskontrolle abgebrochen.

Die Zugriffskontrolle überprüft die Tabelle `vacmAccessTable` mit den Parametern `groupName`, `contextName`, `securityModel` und `securityLevel`. Falls kein Eintrag in der Tabelle gefunden wurde, wird der Fehler `noAccessEntry` zurückgegeben und die Zugriffskontrolle abgebrochen.

Die Zugriffskontrolle untersucht daraufhin, ob der Eintrag in der `vacmAccessTable` Tabelle eine Referenz auf einen MIB-Sicht vom Typ `viewType` enthält. Der Parameter viewType gibt die Zugriffsart an und kann die Werte `read/write/notify` annehmen. Falls der Eintrag keine entsprechende MIB-Sicht definiert wird der Fehler `noSuchView` zurückgegeben und die Zugriffskontrolle abgebrochen.

Die Zugriffskontrolle benutzt den im vorhergehenden Schritt gefunden MIB-View Namen als Index für die `vacmViewTreeFamilyTable` Tabelle. Falls kein Eintrag gefunden werden konnte wird der Fehler `noSuchView` zurückgegeben und die Zugriffkontrolle abgebrochen.

Im letzten Schritt überprüft die Zugriffskontrolle, ob die Objektinstance `variableName` in der gefundenen View enthalten ist. Wenn ja wird der Wert `accessAllowed` zurückgeben, andernfalls der Fehler `notInView`.

# 9 Agenten für das Netzmanagement

## 9.1 Einführung

Gemäß dem Manager-Agent Modell muss auf jedem zu verwaltenden Knoten ein SNMP-Agent installiert werden, der über das SNMP-Protokoll mit der Managementstation kommuniziert.

Mit der Vielzahl von verschiedenen Netzknoten gibt es eine ebenso große Auswahl von Agenten zur Verwaltung dieser Knoten. In den meisten Fällen stellen Hersteller von netzfähigen Rechnern oder Netzkoppelelementen bereits SNMP-Agenten zur Verfügung, die zumindest die **MIB-II** unterstützen. So sind zum Beispiel alle Workstations von SUN standardmäßig mit einem MIB-II SNMP Agenten ausgerüstet und können so von jedem SNMP-fähigen Manager verwaltet werden.

Für spezielle Netzknoten, wie z.B. Router, liefern die Hersteller in der Regel SNMP-fähige Agenten, die neben der MIB-II über eine **private MIB** verfügen, so dass auch herstellerspezifische Eigenschaften des jeweiligen Netzknotens verwaltet werden können. Tatsächlich bieten viele Hersteller nicht nur entsprechende Agenten sondern auch die dazugehörige Managementsoftware an, die dann in Management Plattformen, wie z.B. HP OpenView, integriert werden können.

Oftmals genügen die von einem Standard-Agenten zurückgelieferten Informationen nicht den speziellen Anforderungen. Für diesen Fall stehen dem Anwender sowohl professionelle als auch frei verfügbare **Entwicklungsumgebungen** für die Implementierung von eigenen SNMP-Agenten zur Verfügung. Allen Entwicklungsumgebungen ist gemeinsam, dass die Kommunikation zwischen SNMP-Manager und Agent dem Anwender verborgen bleibt, so dass sich dieser nur um die Implementierung der eigentlichen Funktionalität kümmern muss.

## 9.2 Proxy-Agenten

Mit der Einführung von SNMP wurde es möglich, TCP/IP basierende Rechnernetze zu verwalten. Netzknoten, die allerdings den vollständigen TCP/IP Protokollstack nicht unterstützen (z.B. einfache Netzkoppelelemente wie Repeater oder Bridges), sind vom Netzmanagement zunächst ausgeschlossen.

Für diesen Fall behilft man sich mit sogenannten **Proxy-Agenten**. Mit Hilfe eines Proxy-Agenten können Netzelemente verwaltet werden, die nicht in der Lage sind, direkt mit der Netzmanagementstation (NMS) zu kommunizieren. Soll ein solches Gerät verwaltet werden, sendet die NMS die Anfragen an den Proxy-Agenten und gibt an, welches Gerät eigentlich adressiert werden sollte. Der Proxy-Agent übersetzt die Protokollaufrufe, die er von der NMS empfängt, in die vom fremden Gerät unterstützten Aufrufe. Diese Umsetzung kann auf zwei verschiedene Arten erfolgen:

- Das fremde Gerät unterstützt das Management-Protokoll, aber nicht den End-zu-End Dienst für die Übertragung. Dann muss der Agent lediglich die Protokoll-Dateneinheiten herausfiltern und mit einem anderen Übertragungsprotokoll an das fremde Gerät schicken.

- Das fremde Gerät unterstützt ein anderes Management-Protokoll. Der Proxy-Agent muss das von der NMS verwendete Protokoll übersetzen, er fungiert als **Anwendungs-Gateway**.

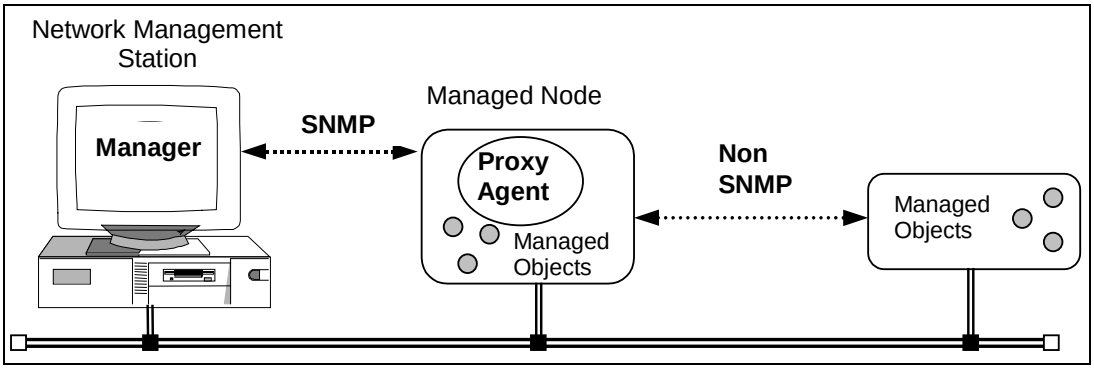



*Bild 9-1* Konzept des Proxy-Agenten

Eine weitere interessante Anwendung für einen Proxy-Agenten ist das Zwischenspeichern von Management-Informationen. Dies ist für Umgebungen interessant, in denen bestimmte Management-Anfragen sehr häufig gestellt werden. Falls sich die abgefragten Informationen mit einer geringeren Frequenz ändern, kann der Proxy-Agent die Anfragen beantworten. Auf diese Weise begrenzt man zum einen den Verkehr zwischen dem Proxy-Agent und dem verwalteten Knoten und zum anderen den Rechenaufwand in den verwalteten Knoten.

## 9.3 Erweiterbare SNMP-Agenten

Auf jeden zu verwaltenden Knoten muss gemäß dem Internet-Netzmanagement Rahmenwerk genau ein Agent installiert sein, der dann mit der Managementstation kommuniziert. Es ist zunächst nicht möglich, mehrere Agenten auf einem Knoten zu installieren, da alle SNMP Agenten denselben Port 161 für sich beanspruchen.

Dies hat zur Folge, dass die gesamte Management-Funktionalität in einem einzigen Agenten (**monolithischer Agent**) implementiert werden muss. Sehr bald erkannte man, dass die Implementierung monolithischer Agenten in Bezug auf die Erweiterbarkeit äußerst unzureichend ist.

### 9.3.1 Das Master-Agent Konzept

Nach diesem Konzept gibt es auf dem zu verwaltenden Knoten einen **Master SNMP-Agenten**. Dieser Master-Agent allein kommuniziert über SNMP mit der Netzmanagementstation. Des weiteren verwaltet der Master-Agent einen oder auch mehrere **Subagenten**. Jeder dieser Subagenten wiederum verwaltet eine eigene private Management Information Base, siehe Bild 9-2.

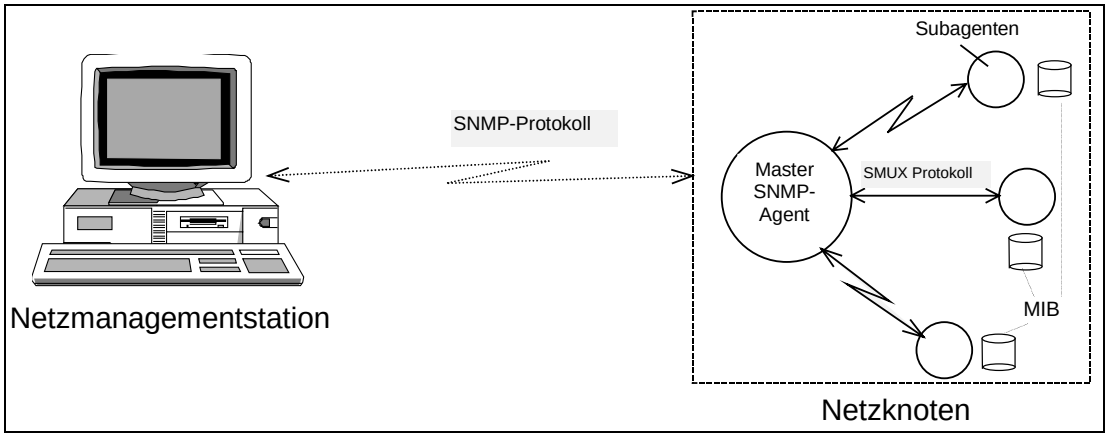



*Bild 9-2 Master-Agent und Subagenten*

Beim Systemstart melden sich alle Subagenten beim Master-Agenten an und registrieren die MIB, die sie verwalten. Damit besitzt der Master-Agent sämtliche Informationen, die er benötigt um seine Subagenten verwalten zu können.

Empfängt der SNMP-Agent eine Anfrage von der Managementstation, die sich auf ein Managed Object innerhalb des Subagenten bezieht, leitet der SNMP-Agent die Anfrage einfach an diesen Subagenten weiter. Der Subagent ist nun dafür verantwortlich die gewünschte Information zu beschaffen und an den SNMP-Agenten zurückzusenden, der diese dann an die Managementstation weiterleitet.

Umgekehrt kann ein Subagent **Traps** an den Master-Agenten senden, der diese wiederum an eine oder mehrere Managementstationen durchreicht.

Mit dieser Architektur ist es nun möglich, einen SNMP-Agenten beliebig zu erweitern, indem zusätzliche Subagenten zur Verfügung gestellt werden. Die Kommunikation zwischen Agent und Netzmanagementstation wird dabei vollkommen von dem Master-Agenten übernommen.

Für die Kommunikation zwischen Master-Agent und Subagent wurde das SMUX (SNMP Multiplexing) Protokoll in RFC1227 definiert. Das SMUX-Protokoll ist sehr einfach gehalten und definiert zusätzlich zu den SNMP-Paketen noch fünf weitere PDUen (Protocol Data Units) zur Kommunikation zwischen Master- und Subagent. Ein Großteil der professionellen wie auch frei verfügbaren Entwicklungsumgebungen für SNMP-Agenten basieren auf dem Prinzip des erweiterbaren SNMP-Agenten.

# 9.4 Solstice Enterprise Agents

## 9.4.1 Einführung

Für die Entwicklung erweiterbarer SNMP Agenten gibt es eine Vielzahl von frei erhältlichen wie auch professionellen Entwicklungsumgebungen. Auch die Firma SUN bietet in ihrer neuesten UNIX-Betriebsystemversion Solaris 2.6 eine komplette Entwicklungsumgebung für SNMPv1 Agenten an[5].

Die Architektur der **Solstice Enterprise Agents** (SEA) von SUN basiert auf dem bereits vorgestellten Master-Agenten Modell. Demnach gibt es auf einem zu verwaltenden Netzknoten genau einen Master-Agenten, der für die Kommunikation mit dem SNMP Manager verantwortlich ist.

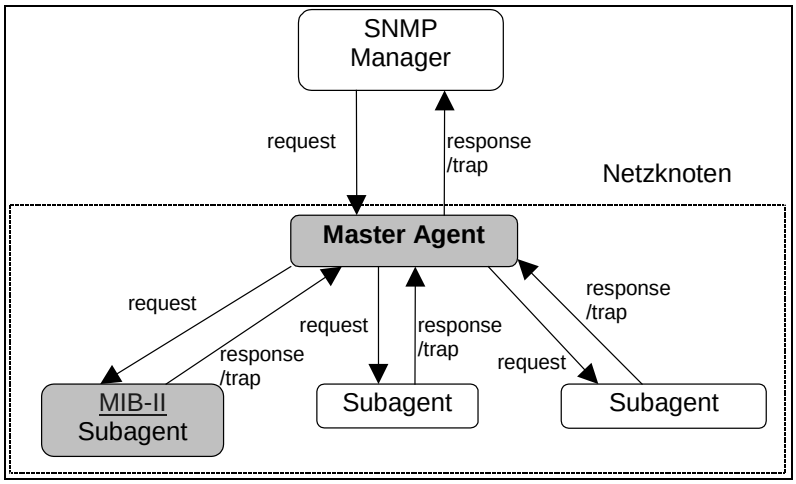



*Bild 9-3 Architektur der Solstice Enterprise Agents*

Des weiteren kann es mehrere Subagenten geben, die sich beim Master-Agenten anmelden. Sobald eine Anfrage vom Manager für einen der Subagenten vorliegt, leitet der Master-Agent die Anfrage entsprechend weiter. Der Master-Agent übernimmt damit die Aufgabe eines Multiplexers.

Die SEA-Laufzeitumgebung beinhaltet den Master-Agenten sowie ein Subagent, der die MIB-II implementiert.

Es ist ebenfalls möglich bereits bestehende SNMP Agenten (engl. legacy agents) als Subagenten in diesem Konzept zu integrieren. Voraussetzung ist allerdings, dass der Port auf dem der SNMP Agent lauscht, veränderbar ist.

[5] Die Laufzeitumgebung für Solstice Enterprise Agents ist bereits vorinstalliert. Eine vollständige Entwicklungsumgebung kann kostenlos von der SUN Internet Home-Page geladen werden.

## 9.4.2 Der Master-Agent

Der Master-Agent ist der Kern der SEA Technologie von SUN. Er läuft als Daemon Prozess und lauscht auf UDP Port 161 auf Anfragen von einem SNMP Manager. Um Traps von seinen Subagenten zu empfangen, öffnet der Master-Agent noch einen weiteren Port 162. Diese Traps können dann anschließend an verschiedene Manager (einstellbar über eine Konfigurationsdatei) verteilt werden.

Der Master-Agent wird automatisch über ein Startup-Skript beim Booten des Systems gestartet. Beim Aufruf liest der Master-Agent verschiedene Konfigurationsdateien und führt daraufhin geeignete Aktionen durch (z.B. Start und Registrierung von Subagenten).

## 9.4.3 Aufruf und Registrierung der Subagenten

Ein Subagent kann auf zwei verschiedene Arten gestartet und beim Master-Agenten registriert werden.

### 9.4.3.1 Statischer Aufruf und Registrierung

Der Master-Agent startet alle Subagenten deren Ressourcen-Dateien (siehe 9.5.1) im SNMP-Konfigurationsverzeichnis `/etc/snmp/conf` stehen. Falls der Master-Agent einen Agenten erfolgreich starten kann, erzeugt er einen entsprechenden Eintrag in einer internen Tabelle. Die Information, für welche Managed Objects sich der Subagent registrieren möchte, liest der Master-Agent dabei aus einer Registrierungsdatei (siehe 9.5.2) des Subagenten. Damit ist der Subagent registriert und bereit SNMP Anfragen vom Master-Agenten zu empfangen.

### 9.4.3.2 Dynamischer Aufruf und Registrierung

Es ist ebenfalls möglich Agenten, für die keine Ressourcen-Datei existiert, manuell oder aber über ein Startup-Skript automatisch beim Booten zu starten. Die Subagenten können sich damit dynamisch beim Master-Agenten registrieren.

In diesem Fall sendet der Subagent beim Hochfahren ein `SET`-Request an den Master-Agenten. Dieser `SET`-Request enthält sämtliche, für die Registrierung notwendigen Informationen. Die dynamische Registrierung von Subagenten ist jedoch nur für Agenten möglich, die mit der SEA Entwicklungsumgebung implementiert wurden.

## 9.4.4 Kommunikation mit den Subagenten

Die Kommunikation zwischen Master-Agent und Subagenten erfolgt über ein eigen entwickeltes Protokoll (und nicht über das SMUX–Protokoll). Als Transportprotokoll kommt das User Datagram Protocol (UDP) zum Einsatz). Für jeden Subagent benutzt der Master-Agent dabei einen unterschiedlichen UDP-Port.

`GET` und `GET-NEXT` Anfragen werden vom Master-Agenten direkt an die entsprechenden Subagenten gesendet.

`SET` Anfragen werden in mehreren Schritten realisiert. Zunächst werden die Werte aller Variablen, die gesetzt werden sollen, über einen `GET`-Request abgefragt.

Anschließend sendet der Master-Agent an alle betroffenen Subagenten einen SET-Request. War der SET Aufruf erfolgreich, wird eine SUCCESS Antwort an den Manager zurückgeschickt. War der SET Aufruf nicht erfolgreich, so sendet der Master-Agent einen weiteren Set-Request mit den ursprünglichen Werten an die betroffenen Subagenten und liefert einen Fehlerwert an den Manager zurück.

Traps werden von den Subagenten an den Master-Agenten gesendet. Der Master-Agent entscheidet daraufhin, welche Manager diesen Trap empfangen sollen. Diese Entscheidung ist über Dateien konfigurierbar.

## 9.5 Konfiguration von Solstice Enterprise Agents

Master- und Subagenten können über verschiedene Dateien konfiguriert werden. Insgesamt stehen die folgenden Dateien für eine Konfiguration zur Verfügung:

- Ressourcen-Konfigurationsdatei (engl. resource configuration file)
- Registrierungsdatei (engl. registration file)
- Zugriffskontrolldatei (engl. access control file)

Im folgenden sollen die verschiedenen Konfigurationsmöglichkeiten kurz beschrieben werden. Eine vollständige Beschreibung dieser Dateien befindet sich im Benutzerhandbuch des Solstice Enterprise Agents.

### 9.5.1 Ressourcen-Konfigurationsdatei

Für jeden Subagent, der automatisch vom Master-Agent gestartet werden soll, muss eine Ressourcen-Datei angelegt werden. Diese Dateien werden dabei ausschließlich vom Master-Agent benötigt und beim Startup eingelesen.

Eine Ressourcen-Datei für einen Subagenten definiert in einer **resource** Gruppe Informationen zum Starten des Subagenten durch den Master-Agenten, zum Beispiel:

```
resource =
{
   {
      type              = "dynamic"
      registration_file = "/etc/snmp/conf/my_agent.reg"
      command           = "/home/landwehr/SNMP/bin/my_agent
                             -p $PORT"
   }
}
```

Die Variablen in der **resource** Gruppe beziehen sich nur auf den Subagenten. Die wichtigsten Variablen in dieser Gruppe sind:

`type` .. enthält den Wert „`dynamic`“ für SEA-Agenten und den Wert „`legacy`“ für alle anderen Agenten.

`registration_file` .. enthält den Namen und Pfad der Registrierungsdatei für den Subagenten.

`command` .. enthält den Namen und Pfad des ausführbaren Agenten.

Für bereits existierende SNMP-Agenten, die nicht mit Hilfe der SEA-Entwicklungsumgebung implementiert wurden, muss eine solche Ressourcen-Datei definiert werden.

### 9.5.2 Registrierungsdatei

Für jeden Subagenten muss eine Registrierungsdatei definiert werden. In dieser Registrierungsdatei werden die für den jeweiligen Agenten spezifischen Informationen festgelegt. Ein Beispiel einer solchen Datei ist im folgenden dargestellt:

```
macros =
{
   demoMib     = experimental.1.15.1.1
}


agents =
{
   {
      name           = "MyAgent"
      subtrees       = { demoMib }
      timeout        = 20000       # in micro secs
      watch_dog_time = 300         # in seconds
   }
}
```

Diese erste Gruppe `macros` definiert Makros für die Verwendung in der `agents` Gruppe. Die wichtigsten Variablen, die in der `agents` Gruppe verwendet werden, sind:

`name`.. gibt dem Agenten einen eindeutigen Namen.

`subtrees`.. enthält eine Liste von Object Identifiers für Unterbäume, die vom Agenten verwaltet werden. Es können auch mehrere Unterbäume angegeben werden.

`timeout`.. jeder Agent kann eine eigene `timeout`-Zeit für noch ausstehende Anfragen angeben.

`watch_dog_time`.. der Master Agent benutzt diese `timeout`-Zeit, um zu überprüfen, ob der Subagent noch aktiv ist.

### 9.5.3 Zugriffskontrolldatei

Diese Konfigurationsdatei regelt die Zugriffsrechte des Agenten und kann für jeden Subagenten wie auch für den Master-Agenten definiert werden. Die Zugriffskontrolldatei enthält die zwei Gruppen `acl` und `trap`.

Die erste Gruppe `acl` regelt den Zugriff auf einen Subagenten über Community-Namen. Hier wird festgelegt, welche Community-Namen der Agent akzeptiert, welche Zugriffsberechtigungen für diese Community-Namen existieren und von welchen Managerstationen auf den Agenten zugegriffen werden darf.

```
acl =
{
   {
      communities = public
      access      = read-only
      managers    = hubble, snowbell
   }
}
```

In der zweiten Gruppe `trap` werden Informationen für die Zustellung von Traps hinterlegt. Hier wird zum einen der Community-Namen für Traps festgelegt, sowie die Namen der Managerstationen, an die Traps gesendet werden sollen.

```
trap =
{
   {
      trap-community = SNMP-trap
      hosts = hubble, snowbell
      {
         enterprise = "Sun"
         trap-num = 1, 2-5
      }
   }
}
```

Durch Angabe von Enterprise-ID und Trap-Nummer ist es möglich, nur eine Untermenge an Traps an die entsprechenden Managerstationen weiterzuleiten.

## 9.6 Entwicklung eines SEA-Subagenten

Die SEA-Entwicklungsumgebung enthält als wichtigste Bestandteile einen MIB-Compiler/Code-Generator und zwei SNMP-Bibliotheken. Ein umfangreiches Benutzer/Referenzhandbuch sowie Beispieldateien und -implementierungen erleichtern den Einstieg für die Implementierung eines eigenen Subagenten.

Im nun folgenden Abschnitt sollen die einzelnen Schritte zur Implementierung eines „Solstice Enterprise Agents“ beschrieben werden.

### 9.6.1 Spezifikation der MIB

Der erste Schritt vor der eigentlichen Implementierung eines Agenten besteht darin, zu spezifizieren, welche Daten der Agent liefern soll, von welchem Typ diese Daten sind und welche Zugriffsrechte für diese Daten gelten.

Diese Informationen werden formal in einer eigenen **MIB** (Management Informationen Base) spezifiziert (siehe 6.3.2).

### 9.6.2 MIB-Compiler und Code-Generator

Die Kommunikation zwischen Master- und Subagenten wird von der SEA-Entwicklungsumgebung gekapselt und ist somit für den Entwickler völlig transparent. Dazu werden die in der MIB des Subagenten beschriebenen Informationen mit Hilfe des MIB-Compilers und Code-Generators „`mibcodegen`“ direkt in C-Code umgewandelt.

Der MIB-Compiler „`mibcodegen`“ steht unter dem Verzeichnis `/usr/bin` und wird mit den folgenden Parameter aufgerufen:

```
mibcodegen –b <subagent> -f <mib-file1> [<mib-fileN>] [–h]
```

Über den Parameter `–b` wird der Name des Subagenten festgelegt. Dieser Name muss eindeutig sein und mit dem Eintrag in der Registrierungsdatei (siehe Abschnitt 9.5.2) übereinstimmen.

Über den zweiten Parameter `–f` kann nun eine (oder auch mehrere) MIB angegeben werden. Der MIB-Compiler überprüft zunächst die Syntax der übergebenen MIB. Wurden keine Fehler in der MIB festgestellt, generiert der MIB-Compiler verschiedene C-Dateien (engl. stub-files), die dem Entwickler als Schablonen für die weitere Entwicklung des Subagenten dienen.

Neben der vom Entwickler definierten MIB liest `mibcodegen` automatisch Definitionen aus einer System-MIB `/var/snmp/mibs/mib-core.txt`. Insgesamt werden die folgenden stub-files generiert:

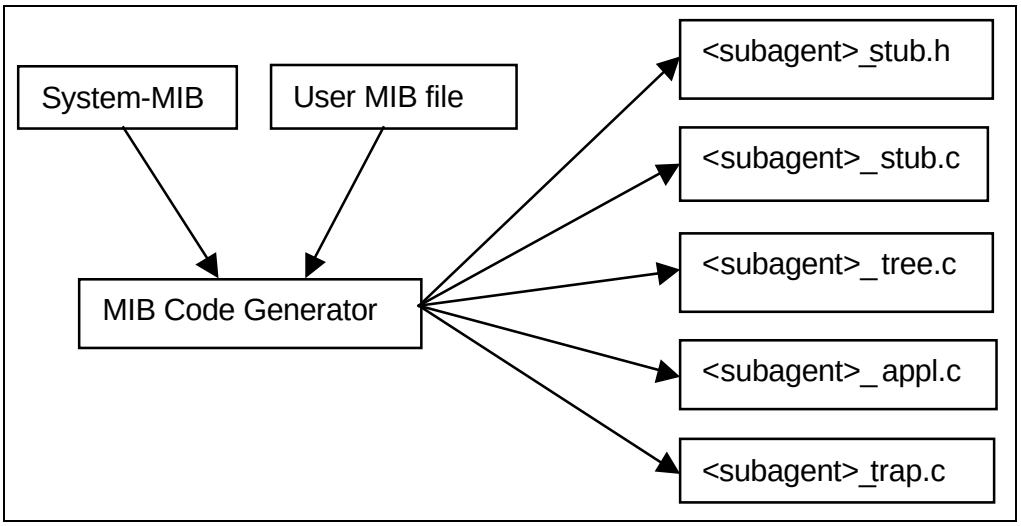



*Bild 9-4 MIB-Compiler und Code-Generator*

Der Bezeichner `<subagent>` in den Namen der generierten Dateien entspricht dem Namen des Subagenten der als Parameter übergeben wurde. Die folgende Tabelle gibt eine Übersicht über alle generierten Quelldateien und deren Bedeutung:

| | |
|---|---|
| `<subagent>_stub.h` | enthält die Deklaration von Datenstrukturen in der MIB. Diese Datei sollte nicht modifiziert werden. |
| `<subagent>_stub.c` | enthält Callback-Funktionen zum Lesen und Schreiben (sofern entsprechende Zugriffsrechte existieren) für jedes Managed Object in der MIB. Diese Callback Funktionen müssen vom Entwickler mit anwendungsspezifischen Code „gefüllt“ werden. |
| `<subagent>_tree.c` | enthält die Logik zum Aufruf der entsprechenden GET- und SET-Callback-Funktionen in der `stub.c` Datei. Diese Datei sollte nicht modifiziert werden. |
| `<subagent>_appl.c` | enthält allgemeine Callback-Funktionen, die vom Entwickler an die entsprechenden Bedürfnisse angepaßt werden können. |
| `<subagent>_trap.c` | enthält Logik für die Behandlung von Traps. Diese Datei sollte nicht modifiziert werden. |

*Tabelle 9-1 Generierte C-Dateien des MIB-Compilers*

Enthält die User-MIB Tabellen, so erzeugt der MIB-Compiler für jede Tabelle in der MIB eine eigene Quelldatei. Für weitere Informationen über die Behandlung von Tabellen sei auf das User-Manual verwiesen.

Im folgenden Abschnitt werden die erforderlichen Anpassungen an den generierten Dateien beschrieben.

## 9.6.3 Anpassungen der generierten Dateien

Die vom MIB-Compiler generierten Dateien enthalten Callback-Funktionen, die vom Entwickler an die eigenen Bedürfnisse angepasst werden müssen. Um das „Ausfüllen“ dieser Callback-Funktionen zu erleichtern enthalten diese teilweise bereits Standardimplementierungen.

### 9.6.3.1 Anpassungen der <subagent>_appl.c Datei

Diese Datei enthält allgemeine Funktionen zur Steuerung des Subagenten. Zu diesen Funktionen gehören:

| | |
|---|---|
| `void agent_init ()` | Diese Funktion wird beim Start des Subagenten aufgerufen und kann vom Entwickler zum Beispiel zur Initialisierung von Daten benutzt werden. |
| `void agent_end ()` | Diese Funktion wird aufgerufen, falls der Subagent ein `SIGTERM` Signal bekommt und kann vom Entwickler für „Aufräumarbeiten“ verwendet werden. |
| `void agent_loop ()` | Diese Funktion wird in regelmäßigen Abständen aufgerufen und kann z.B. für die periodische Überprüfung von Schwellwerten eingesetzt werden. |
| `void main ()` | Die `main` Funktion enthält als einzige Anweisung den Aufruf der Funktion `SSAMain ()`, die den Subagenten initialisiert und startet. Diese Funktion muss immer zuletzt innerhalb der `main` Funktion aufgerufen werden. |

*Tabelle 9-2 Funktion der <subagent>_app.c Datei*

Ist der Subagent einmal gestartet, läuft er in einer Endlosschleife und kann nur über ein UNIX-Signal beendet werden.

### 9.6.3.2 Anpassungen der <subagent>_stub.c Datei

Diese Datei enthält für jedes Managed Object der „User-MIB“ mit Lese- und Schreibzugriff eine entsprechende `get_<variable>()` und `set_<variable>()` Callback-Funktion. Für Managed Objects, auf die nur lesend zugegriffen werden darf, wird nur die entsprechende `get_<variable>()`Funktion generiert.

Die Bezeichnung `<variable>` ist der Platzhalter für den Namen des jeweiligen Managed Objects. Für ein Managed Object `demoInteger` mit Lese- und Schreibzugriff werden zum Beispiel die folgenden zwei Funktionen erzeugt:

```
int get_demoInteger (int* demoInteger) {}
int set_demoInteger (int pass, int* demoInteger) {}
```

Es ist nun die Aufgabe des Entwicklers beide Funktionen entsprechend ihrer Bedeutung zu vervollständigen. Der MIB-Compiler generiert dabei bereits einen Beispielcode für jede Funktion, der im einfachsten Fall nur noch entsprechend angepasst werden muss.

### 9.6.4 Compilieren und Linken des Subagenten

Als letzten Schritt müssen nun die generierten Dateien, sowie eventuell eigene Dateien, compiliert werden. Beim anschließenden Linken müssen die beiden SEA-Bibliotheken `libssasnmp` und `libssagent` hinzugebunden werden.

Die Bibliothek `libssasnmp` stellt dabei low-level SNMP Schnittstellenfunktionen zur Verfügung, während die Bibliothek `libssagent` high-level Funktionen bereitstellt, um SNMP-Anfragen zu empfangen, entsprechende Antworten zurückzuliefern und Speicher für die Parameter von SNMP-Anfragen zu allokieren.

Nach erfolgreichem Linken, müssen nun (wenn nicht schon geschehen) die in Kap. 9.5 vorgestellten Konfigurationsdateien angelegt werden. Die Registrierungsdatei sowie die Zugriffskontrolldatei muss für jeden Subagenten definiert werden. Die Ressourcen-Konfigurationsdatei ist optional, falls der Subagent nicht durch den Master-Agenten beim Systemstart hochgefahren werden soll (dynamische Registrierung).

Anschließend kann der Subagent direkt aufgerufen werden, wobei die folgenden Parameter gesetzt werden können:

```
<subagent> -c <registration file> -a <control-access file>
           -d <debug-level>
```

Damit registriert sich der Subagent automatisch beim Master-Agent und steht nun für SNMP-Anfragen zur Verfügung. Die beiden Parameter für die Registrierungsdatei bzw. für die Zugriffskontrolldatei müssen nur angegeben werden, falls sich beide Dateien nicht im Standard-Konfigurationsverzeichnis `/etc/snmp/conf` befinden.

Über die `-d` Option kann der Grad (von 0 - 4) der vom Subagenten ausgegebenen Debug-Informationen angegeben werden.

### 9.6.5 Zusammenfassung

Mit der SEA-Entwicklungsumgebung können auf recht einfache Weise SNMPv1 fähige Agenten implementiert werden. Mit Hilfe des MIB-Compilers, der die MIB-Definitionen direkt in C-Quellcode übersetzt, wird die Kommunikation zwischen Master-Agent und Subagenten vollständig verborgen.

Die Aufgabe des Entwicklers besteht (neben der Definition der MIB) nur noch darin, die generierten C-Dateien für seine Bedürfnisse anzupassen, sowie die `get/set` Callback-Funktionen für alle in der MIB definierten Managed Objects zu implementieren.

# 10 Begriffsverzeichnis

# A

**Address Resolution Protocol** — Ein Protokoll der Internet-Protokollfamilie zur Abbildung von IP-Adressen auf Ethernet-Adressen.

**Agent** — SNMP-Managementprozess, der auf dem zu verwaltenden Rechner abläuft, Informationen über den Rechner sammelt und an die →Netzmanagementkonsole weiterleitet.

**Anwendungsschicht** — siehe →Application Layer

**Anwendungsschicht (TCP/IP)** — Oberste Schicht im TCP/IP-Architekturmodell. Sie ist verantwortlich für die Verwaltung der Kommunikation zwischen Anwendungsprozessen.

**Anycast-Adresse** — IPv6-Adresse, mit der mehrere Interfaces, meist verschiedener Knoten angesprochen werden können. Das Paket wird nur an einen Empfänger ausgeliefert.

**Application Layer** — Anwendungsschicht, siebte Schicht des ISO/OSI-Modells. Sie stellt die Schnittstelle des Kommunikationssystems zu Anwendungsprogrammen dar. Die Schicht 7 enthält allgemeine Hilfsdienste für die Kommunikation sowie spezielle Kommunikationsdienste wie File Transfer, Virtuelles Terminal etc.

**ARPA** — siehe →Defense Advanced Research Projects Agency.

**ARP** — siehe →**A**ddress **R**esolution **P**rotocol

**ASN.1** — **A**bstract **S**yntax **N**otation One. OSI-Sprache für die Beschreibung einer abstrakten Syntax. ASN.1 ist die Syntax, mit der eine →MIB beschrieben wird. Durch diese Beschreibung wird einer →Netzmanagementkonsole mitgeteilt, welche Informationen in der MIB eines Agenten zur Verfügung stehen.

# B

**Bitübertragungsschicht** — siehe →Physical Layer

**Bridge** — Ein auf Schicht 2 des →OSI-Modells arbeitendes Netzkoppelelement, das zwei oder mehr →LANs gleicher Topologie miteinander verbindet.

**Broadcasts** — Broadcasts sind spezielle Datenpakete in einem Netz, die an alle empfangsbereiten Stationen gerichtet sind.

**Broadcast Address** — Eine Adresse, die alle Stationen auf diesem Medium adressiert.

# C

**CCITT** — **C**omité **C**onsultatif **I**nternational de **T**éléphonique et **T**élégraphique. Der internationale Ausschuss von Fernmeldeverwaltungen und –gesellschaften zur Ausarbeitung von Normungsvorschlägen, inzwischen in der ITU aufgegangen. Dieses Gremium arbeitet Empfehlungen für Standards in den Bereichen Datenübertragungs-Geräte-Schnittstellen, Kommunikationsprotokolle, u.a. aus.

**Client** — Anwendung, die Dienste von Servern in Anspruch nehmen kann.

**Client/Server** — Architektur für die Datenverarbeitung mit auf mehreren Rechnern verteilten Anwendungen.

**CMIP** — **C**ommon **M**anagement **I**nformation **P**rotocol, OSI Protokoll für das Netzmanagement.

**CMOT** — **CM**IP **O**ver **T**CP. Spezifikation für den Transport des CMIP-Protokolls über TCP/IP, damit CMIP für die Verwaltung von TCP/IP-Netzwerke benutzt werden kann,

**CSMA/CD** — **C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **D**etection = Vielfachzugriff mit Leitungsabfrage und Kollisionserkennung. Bei Anwendung dieses Zugangsverfahrens prüfen die Stationen von Beginn der Übertragung, ob der Kanal frei ist. Wird eine Kollision festgestellt, weil zwei Stationen gleichzeitig senden wollen, wird die Sendung von beiden beendet. Ein erneuter Sendeversuch erfolgt nach einer zufälligen Wartezeit.

# D

**Daemon** — Ein im Hintergrund laufender Prozess, der bei Bedarf aktiv wird, und nach Erledigung seines Auftrags wieder in den ursprünglichen Zustand zurückkehrt. Die meisten Daemon-Prozesse werden während des Systemstarts gestartet.

**Darstellungsschicht** — siehe →Presentation Layer

**Datagramm** — Eine in sich abgeschlossene Nachricht. Sie wird unabhängig von anderen Datagrammen übertragen.

**Data Link Layer** — Sicherungsschicht, zweite Schicht des ISO/OSI-Modells. Protokolle, die auf dieser Schicht arbeiten stellen eine gesicherte Punkt-zu-Punkt-Verbindung zwischen 2 Stationen zur Verfügung.

**DCE** — **D**ata **C**ommunication **E**quipment, ein Rechner im Netz (Netzknoten), der zur Übermittlung dient.

**DEE** — **D**aten-**E**nd-**E**inrichtung, ein System am Netzrand, das in der Lage ist, mit anderen Endsystemen über das Netzwerk zu kommunizieren.

**Default Route** — Ein Eintrag in der Routing-Tabelle, der immer dann beim Senden eines Datagramms benutzt wird, wenn kein spezieller Eintrag für die Zieladresse in der Routing-Tabelle vorhanden ist.

**Defense Advanced Research Projects Agency (DARPA)** — Eine Agentur des U.S. Verteidigungsministeriums, die Forschungsprojekte fördert. Die Internet-Protokollfamilie wurde unter Mithilfe der DARPA entwickelt. DARPA war früher unter dem Namen →ARPA, der Advanced Research Projects Agency bekannt, die das ARPANET aufbaute.

**Direct Routing** — Darunter versteht man das Senden von IP-Datagrammen zu Empfängern, die auf dem gleichen IP-Netzwerk (oder IP-Subnetzwerk) liegen wie der Sender.

**DNS** — siehe →Domain Name Service

**Domain Name Service** — Namensdienst zur Abbildung von logischen Rechnernamen auf IP-Adressen

**DÜE** — **D**aten-**Ü**bertragungs-**E**inheit, siehe →DCE

# E

**Entity** — Begriff aus der OSI-Terminologie: Prozess in einer Schicht, der die jeweiligen Schichtaufgaben erfüllt. In derselben Schicht eines Endsystems können mehrere Instanzen vorhanden sein.

**Ethernet** — Ein Ethernet ist ein Bussystem mit CSMA/CD-Zugriff und Basisbandübertragung. 1979 wurde dieses lokale Netz von den Firmen DEC, Intel und Xerox entwickelt. Als eines der ersten →LANs wurde es zum De-Facto-Standard und vom IEEE (**I**nstitute of **E**lectronical and **E**lectronics **E**ngineers) als Standard übernommen (Norm 802.3). Die Übertragung erfolgt auf Koaxial-, Twisted Pair-, Lichtwellenleiter oder anderen Übertragungsmedien mit 10 Mbit/Sekunde.

# F

**File Transfer Protocol (FTP)** — Das Anwendungsprotokoll, dass einen Dateidienst in der Internet-Protokollfamilie anbietet.

**Fragment** — Ein IP-Datagramm, das nur noch einen Teil der Benutzerdaten eines größeren IP-Datagramms enthält.

**Fragmentation (Fragmentierung)** Der Prozess des Zerlegens eines IP-Datagramms in kleinere Teile, die dann als Ganzes über ein gegebenes physikalisches Medium übertragen werden können.

**Folgeinstanz** Zwei im →OSI-Referenzmodell vertikal benachbarte Instanzen nennt man obere und untere Folgeinstanz.

**Frame** Von der Schicht 2 des →OSI-Referenzmodells erzeugte →PDU

**FTP** siehe →File Transfer Protocol.

# G

**Gateway** Netzkoppelelement, das oberhalb der Schicht 3 des →OSI-Referenzmodells arbeitet und zwei unterschiedliche →LANs oder →WANs miteinander verbindet.

# H

**Hardwareadresse** siehe →Mediaadresse.

**Header** siehe →Protocol Control Information.

**Host** Als Host (engl.: Wirt) werden Rechner bezeichnet, die für andere Einheiten (z.B. Terminals) bestimmte Funktionen wie beispielsweise die Speicherung von Daten übernehmen, bezeichnet auch ein Endsystem im Internet.

**Host-ID** siehe →Hostbezeichner.

**Hostbezeichner** Der Teil einer IP-Adresse, der zu einem Host an diesem IP-Netzwerk gehört.

**Hub** Allgemeine Bezeichnung für Konzentrator, Sternverteiler, Multiport-Repeater.

# I

**IANA** siehe →Internet Assigned Numbers Authority.

**IAB** Das **I**nternet **A**ctivities **B**oard ist ein Ausschuss, der die Entwicklung des Internet und der damit verbundenen Technologien überwacht und steuert. Die von den Gremien der IAB entwickelten Standards werden vom IAB in RFCs publiziert und bei ausreichender Akzeptanz zu Internet Standards erklärt.

**ICMP** siehe →Internet Control Message Protocol.

**Instanz** siehe Entity

**Interface Control Information** Steuerinformation für die Folgeschicht im →OSI-Referenzmodell zur Verarbeitung der übergebenen User-Daten.

**Internet** Ein weltweiter Verbund von mit TCP/IP arbeitenden Rechnernetzen, welche durch Router miteinander verbunden sind und sich wie ein einziges, virtuelles Netzwerk verhalten.

**Internet Assigned Numbers Authority (IANA)** Organisation, welche für die Vergabe von Nummern im Zusammenhang mit der Internet-Protokollfamilie verantwortlich ist. Die IANA vergibt beispielsweise UDP/TCP-Portnummer und IP-Netzwerkadressen

**Internet Control Message Protocol (ICMP)** Ein einfaches Nachrichtenprotokoll für IP.

**Internet Level** siehe →Internetschicht.

**Internet-Protokollfamilie** Die Internet-Protokollfamilie ist eine Anfang der siebziger Jahre vom DoD (Department of Defense, US-Verteidigungsministerium) entwickelte umfangreiche Protokollfamilie zur Verbindung heterogener Wide Area Networks (→WANs). Sie ist derzeit praktisch der Standard für UNIX-basierende oder heterogene Netze.

**Internetschicht** Die Schicht in der Internet-Protokollfamilie, die dafür verantwortlich ist, dass sowohl der Aufbau als auch das Übertragungsmedium in jedem physikalischen Netzwerk unsichtbar bleiben

**IP** **I**nternet **P**rotocol. Das Schnittstellen-Protokoll, das in der Internet-Protokollfamilie einen verbindungslosen Dienst zur Verfügung stellt.

**IP-Adresse** Eine 32-Bit-Einheit, die einen Anschlusspunkt in einem Internet bezeichnet.

**ISO** **I**nternational **S**tandards **O**rganization, eine internationale Organisation, die die Entwicklung weltweiter Normen – für alle Sachgebiete – koordiniert und für deren Veröffentlichung sorgt. Ihre Mitglieder sind die nationalen Normungsinstitute wie beispielsweise DIN (Deutschland) oder ANSI (USA).

# J

# K

**Kernel** Kern des Betriebssystems.

**Knoten** Als Knoten wird ein an ein Rechnernetz angeschlossenes Gerät bezeichnet, das Daten empfängt oder sendet. Dieses können einzelne Rechner, Server, Drucker oder Netzkoppeleinheiten sein, die von mehreren Netzteilnehmern angesprochen werden.

**Kommunikations-steuerungsschicht** siehe →Session Layer

**Konzentrator** Der Konzentrator stellt eine intelligente Komponente der Netz-Topologie dar, an die Knoten oder weitere Konzentratoren angeschlossen werden.

# L

**LAN** **L**ocal **A**rea **N**etwork. Laut ISO ist ein lokales Netzwerk „ein innerhalb von Grundstücksgrenzen unter rechtlicher Kontrolle des Benutzers befindliches Netzwerk für die bitserielle Übertragung von Informationen zwischen dessen unabhängigen, miteinander gekoppelten Elementen." Ein lokales Netzwerk ist also ein örtlich stark eingeschränktes Netzwerk, das meistens innerhalb eines Gebäudes oder eines Firmensitzes installiert ist.

**Link-local-Adresse** IPv6-Adresse, die nur auf dem lokalen Transportweg eindeutig ist. Sie kann verwendet werden, um direkt benachbarte Knoten zu adressieren.

# M

**MAC** **M**edia **A**ccess **C**ontrol, Zugriffssteuerung auf das Medium. Von der →ISO definierte Unterebene der Schicht 2 des →OSI-Referenzmodells. Bei →Ethernet-Netzen gehören die Quell- und Zieladresse sowie der Protokolltyp zu den MAC-Layer-Daten.

**MAN** Das **M**etropolitan **A**rea **N**etwork ist ein Rechnernetz innerhalb eines städtischen Bereichs mit der Ausdehnung bis ca. 50 km. Seine Übertragungsgeschwindigkeit liegt, abhängig vom eingesetzten Übertragungsverfahren, zwischen 100 Mbit/s und mehreren GBit/s.

**Managed Object** Die abstrakte Beschreibung einer Systemressource durch die Definition der abfragbaren Daten.

**Manager** Teil eines Management-Systems, welches Informationen der Agenten empfängt und auswertet.

**Map** Graphische Darstellung der Topologie eines Netzes oder Teilnetzes unter →OVW.

**Mediaadresse** Die Adresse einer physikalischen Schnittstelle.

**MIB** — **M**anagement **I**nformation **B**ase. Bezeichnet die Menge der Informationen, die von einem →Agenten abgefragt werden können.

**MIT** — **M**anagement **I**nformation **T**ree. Hierarchischer, baumartig strukturierter Namensraum, der eine eindeutige Identifizierung von in diesem Namensraum angesiedelten Objekten durch Aneinanderreihung der Identifier von der Wurzel bis zum Objekt erlaubt. Dieser Namensraum wird für Managed Objects, FTAM Dokumententypen, RPC Interfacetypen, ASN.1 Module, X.500 Attribute und vieles mehr verwendet.

**MO** — siehe →Managed Object

**Multicast-Adresse** — Adresse, mit der mehrere Interfaces gleichzeitig angesprochen werden können. Das Paket wird an alle Interfaces ausgeliefert.

# N

**Name Server** — Ein Host, auf dem ein Server für den →Domain Name Service läuft.

**Network Layer** — Vermittlungsschicht, dritte Schicht des →OSI-Referenzmodells. Auf dieser Schicht werden End-zu-End-Verbindungen zwischen kommunizierenden Endsystemen realisiert.

**Network Level** — siehe →Schnittstellenschicht

**network-identifier** — Netzwerkbezeichner, der Teil der IP-Adresse, der zu einem Netzwerk im Internet gehört.

**Netz-ID** — siehe →network-identifier.

**Netzknoten** — siehe →DÜE

**NMS** — **N**etwork **M**anagement **S**tation. Station im Netz, auf der eine Management-Software läuft und die das Netz verwaltet.

**NNM** — **N**etwork **N**ode **M**anager. Netzmanagement-Komponente von HP OpenView.

# O

**OID** — **O**bject **Id**entifier, identifiziert ein Element innerhalb des →Management Information Trees (MIT) eindeutig.

**OSIBRM** — **OSI B**asic **R**eference **M**odel siehe →OSI-Referenzmodell

**OSI-Referenzmodell** **O**pen **S**ystem **I**nterconnection. = offene Kommunikationssysteme. Von der →ISO entwickeltes Referenzmodell für Rechnernetze zur Festlegung der Schnittstellen-Standards zwischen Computerherstellern für den Bereich der Hard- und Software-Anforderungen. Dieses Modell teilt die Dienste zur Rechnerkommunikation in sieben Schichten ein, die aufeinander aufbauen.

**OVW** **O**pen**V**iew **W**indows. Oberfläche des Netzmanagement - Tools HP OpenView.

# P

**Partnerinstanz** Zwei im →OSI-Referenzmodell horizontal in einer Schicht kommunizierende Instanzen heißen Partnerinstanzen (peer entities).

**PCI** **P**rotocol **C**ontrol **I**nformation, Protokollsteuerungsinformation. Ein Dienstanbieter fügt zu den ihm übergebenen Nutzdaten einen kleinen Header hinzu, die sogenannte Protocol Control Information. Diese stellt eine Steueranweisung für den Protokollpartner derselben Ebene des anderen Endsystems dar.

**PDU** **P**rotocol **D**ata **U**nit, Protokolldateneinheit. Eine Nachricht die zwischen Partnerinstanzen ausgetauscht wird. Sie besteht aus Nutzdaten und PCI.

**Peer - Partner** →Instanzen der gleichen Schicht in verschiedenen Endsystemen.

**Physical Layer** Bitübertragungsschicht, unterste Schicht des →OSI-Referenzmodells. Sie befasst sich mit der Anbindung eines Endsystems an ein physikalisches Übertragungsmedium und der darauf ablaufenden Bitübertragung.

**Port number** Portnummer, bezeichnet eine Anwendungseinheit durch einen Transportendpunkt des UDP- oder TCP-Protokolls.

**Presentation Layer** Darstellungsschicht, sechste Schicht des →OSI-Referenzmodells. Sie ist verantwortlich für die Strukturierung der Dateneinheiten, die ausgetauscht werden.

**Protokoll** Vorschriften zur Kommunikation zweier Rechner, definieren den Aufbau einer Nachrichtenverbindung und legen das Format der über diese Verbindung ausgetauschten Nachrichten fest.

**Proxy-Agent** Proxy-Agenten übersetzen Protokollaufrufe zwischen Manager- und Agentensystemen im Netzmanagement. Dies wird benötigt, wenn ein Agent nicht das gleiche Management- oder Netzprotokoll wie der Manager unterstützt.

# Q

# R

**Rahmen** siehe →Frame

**RARP** siehe →Reverse Address Resolution Protocol

**Reassemblierung** Der Prozess des Zusammenfügens von Fragmenten am endgültigen Ziel in das ursprüngliche IP-Datagramm.

**Rechner** beinhaltet einen Rechner aus HW, Betriebssystem, Kommunikationssystem sowie Anwendungs- und anderen Systemprogrammen. Durch das Kommunikationssystem ist ein solcher Rechner fähig, über ein Netz mit anderen Rechnern zu kommunizieren. Der Begriff Rechner wird hier äquivalent zu Station und System verwendet. ISO/OSI spricht in abstrakter Weise nur von Systemen, nicht von Rechnern oder Stationen.

**Repeater** Signalverstärker, der den Pegelverlust bei der Übertragung über größere Strecken ausgleicht.

**Reverse Address Resolution Protocol** Ein Protokoll der →TCP/IP-Protokollfamilie. Mit diesem Protokoll werden →MAC-Adressen auf →IP-Adressen abgebildet.

**Router** Ein Router ist ein Gerät zur Verbindung zweier Netze mit gleicher Layer-3-Struktur im →OSI-Referenzmodell und dient zur Kopplung von Netzen unterschiedlicher Topologie.

# S

**SAP** **S**ervice **A**ccess **P**oint, Dienstzugangspunkt. Die Schnittstelle zwischen Schichteninstanz und Dienstbenutzer.

**Schnittstellenschicht** Die Schicht in der Internet-Protokollfamilie, die für die Übertragung über ein einzelnes physikalisches Netzwerk verantwortlich ist.

**SDU** **S**ervice **D**ata **U**nit, Dienstdateneinheit. Nutzdaten einer Schicht, die einer untergeordneten Folgeschicht im Dienstelement übergeben werden.

**Segment** Die Übertragungseinheit in TCP.

**Server** Ein Server ist ein Anbieter von Dienstleistungen, die von einem Client in Anspruch genommen werden. Diese Dienste reichen von grundlegenden Datei- und Druckdiensten bis hin zur Unterstützung von komplexen, verteilten Anwendungen. Viele Netzwerk-Betriebssysteme verfügen über eine Client-Server-Architektur, d.h., ein spezieller, sehr leistungsfähiger Rechner arbeitet als Server, von dem eine große Anzahl von Clients (Arbeitsplatzrechnern) Daten und Programme beziehen.

**Service** Ein Kommunikationsdienst nach ISO/OSI ist eine Dienstleistung, welche auf einem Rechner-Knoten von einer ISO/OSI-Schicht am Dienstzugangspunkt (Service Access Point) dem Aufrufer bereit gestellt wird. Der Kommunikationsdienst wird erbracht durch die Instanzen dieser Schicht und ihr Schichtenprotokoll unter Nutzung der Dienste der tieferen Schichten.

**Service Primitive** Dienstelement. Kommandos zur Inanspruchnahme von Diensten und die Ergebnismeldungen heißen Dienstelemente (service primitives). Künstliche Modellierung, wie ein Dienst von einem Benutzer angefordert oder akzeptiert wird.

**Service Provider** Diensterbringer. Instanz einer untergeordneten Schicht, die den Instanzen einer höheren Schicht am Service Access Point einen Dienst zur Verfügung stellt.

**Service User** Dienstbenutzer. Instanz, die den Dienst einer Instanz der nächst tieferen Schicht nutzt.

**Session Layer** Kommunikationssteuerungsschicht, fünfte Schicht des →OSI-Referenzmodells. Sie regelt die Synchronisation zwischen Prozessen in verschiedenen Systemen.

**SGMP** Das **S**imple **G**ateway **M**onitoring **P**rotocol war die Basis für das →Simple Network Management Protocol.

**Sicherungsschicht** siehe →Data Link Layer

**SNMP** **S**imple **N**etwork **M**anagement **P**rotocol. Das Netz-Management-Protokoll der IETF und De-facto Industrie-Standard für das Netzmanagement.

**SMI** Die **S**tructure of **M**anagement **I**nformation beschreibt die Regeln für die Beschreibung der Objekte, die mit Hilfe eines Netzmanagement-Protokolls erreicht werden können.

**Socket** Ein Paar bestehend aus IP-Adresse und einer Portnummer.

**Source Routing** Der Absender gibt den Pfad zur Zielstation explizit vor.

**Station** siehe →Rechner

**Structure of Management Information** siehe →SMI

**Subnet (Subnetz)** Ein physikalisches Netzwerk innerhalb eines IP-Netzwerks.

**Subnet Mask** Subnetzmaske, eine 32-bit-Größe, die angibt, welche Bits einer IP-Adresse das physikalische Netzwerk angeben.

**Subnet-number** Der Teil einer IP-Adresse, der ein bestimmtes physikalisches Netzwerk innerhalb eines IP-Netzes bezeichnet.

**Subnetting** Das Unterteilen eines IP-Adressraums in mehrere →Subnetze durch Verwendung von IP-Subnetzmasken.

**Subnetwork** Ein Netzwerk, das mehrere Knoten an einem einzigen (virtuellen) Übertragungsmedium miteinander verbindet.

**System** siehe →Rechner

# T

**TCP** **T**ransmission **C**ontrol **P**rotocol, Transportprotokoll, das einen verbindungsorientierten Transportdienst in der Internet-Protokollfamilie anbietet.

**TCP/IP** **T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol. Eine Anfang der siebziger Jahre vom DoD (Department of Defense, US-Verteidigungsministerium) entwickelte umfangreiche Protokollfamilie zur gesicherten Verbindung heterogener Wide-Area-Networks (→WANs). Die beiden Fundamente dieser Protokollfamilie sind das →IP, welches die Schicht 3 des OSI-Modells implementiert, und dessen Analogon →TCP für die vierte Schicht.

**TELNET** Das Anwendungsprotokoll, das einen virtuellen Terminaldienst in der Internet-Protokollfamilie anbietet.

**Token Access** Token Access ist ein kollisionsfreies Zugangsverfahren zur Informationsübertragung in lokalen Netzen (LANs), wird z.B. eingesetzt bei Token Ring, Token Bus und FDDI.

**Transmission Level** siehe →Transportschicht(TCP/IP).

**Transport Layer** Transportschicht, vierte Schicht des →OSI-Referenzmodells. Sie stellt den Prozessen der Endsysteme in transparenter Weise Transportdienste verschiedener Güte zur Verfügung. Transparent bedeutet, dass die Prozesse nicht mehr die Physik des Netzes, sondern nur mehr die Güte des Transportdienstes (Durchsatz, Sicherheit, ...) sehen.

**Transportschicht** siehe →Transport Layer

**Transportschicht (TCP/IP)** Die Schicht, welche für die Datenübertragung zwischen Endsystemen mit einem von der Anwendung gewünschten Grad an Zuverlässigkeit verantwortlich ist. D.h. die Transportschicht bietet einen Ende-zu-Ende-Dienst.

# U

**UDP** **U**ser **D**atagram **P**rotocol, das Transportprotokoll, das einen verbindungslosen Transportdienst in der Internet-Protokollfamilie anbietet.

**Unicast-Adresse** Adresse, die ein einzelnes Interface adressiert.

**UNIX** Ein Betriebssystem für leistungsfähige Microcomputer, Computer und Großrechner, das von AT&T entwickelt wurde.

# V

**Vermittlungsschicht** siehe →Network Layer

# W

**WAN** **W**ide **A**rea **N**etwork, Weitverkehrsdatennetz. Dient zur Übertragung von Daten über größere Entfernungen. Es unterliegt keiner räumlichen Begrenzung und kann kontinental oder innerhalb von Landesgrenzen eingesetzt werden. Es beruht in der Regel auf paketvermittelnden oder leitungsvermittelnden Netzen von öffentlichen und privaten Anbietern (Bsp. Datex-P, Datex-L). Die Übertragungsgeschwindigkeiten der WAN's liegen üblicherweise zwischen 64 kbit/s bis 2,048 Mbit/s.

**well-known Port** Ein durch die IANA dokumentierter Transportendpunkt.

# X

**X.25** Ein verbindungsorientierter Netzwerkdienst.

# 11 Abkürzungen

| | |
|---|---|
| ASCII | American Standard Code of Information Interchange |
| ASN.1 | Abstract Syntax Notation One |
| CCITT | Comité Consultatif International de Téléphonique et Télégraphique |
| CMIP | Common Management Information Protocol |
| CMISE | Common Management Information Service Entity |
| CMOT | Common Management Information Protocol over TCP |
| DNS | Domain Name System |
| DARPA | Defense Advanced Research Projects Agency |
| DoD | Department of Defense |
| EGP | Exterior Gateway Protocol |
| FTP | File Transfer Protocol |
| IAB | Internet Activity Board |
| IANA | Internet Assigned Numbers Authority |
| ICMP | Internet Control Message Protocol |
| IEEE | Institute of Electrical and Electronic Engineers |
| IP | Internet Protocol |
| ISO | International Standardization Organization |
| LAN | Local Area Network |
| MAN | Metropolitan Area Network |
| MIB | Management Information Base |
| MIT | Management Information Tree |
| NCC | Network Controlling Center |
| NIC | Network Information Center |
| NMS | Network Management Station |
| NOC | Network Operating Center |
| OID | Object Identifier |
| OSI | Open Systems Interconnection |
| RFC | Request for Comments |
| SGMP | Simple Gateway Monitoring Protocol |
| SMI | Structure of Management Information |
| SMTP | Simple Mail Transfer Protocol |
| SNMP | Simple Network Management Protocol |
| TCP | Transmission Control Protocol (verbindungsorientiert) |
| UDP | User Datagram Protocol (verbindungslos) |
| UNC | Universal Naming Convention |
| WAN | Wide Area Network |

# 12 Literaturverzeichnis


[1] M. Bosch: Kopplung von Netzen mit unterschiedlichen Protokollarchitekturen; Universität Stuttgart, Institut für Nachrichtenvermittlung und Datenverarbeitung

[2] H. W. Barz: Kommunikation und Computernetze - Konzepte, Protokolle und Standards; Hanser Verlag

[3] A. Zenk: Lokale Netze - Kommunikationsplattform der 90er Jahre; Addison-Wesley

[4] K. Peter: Von LAN zu LAN; PC Magazin Nr.5 vom 24.1.1990

[5] G. Göbel: Brückenschlag zwischen lokalen Netzen; Mini Micro Magazin 7-8/1990

[6] P. Mandl: Wege zum Ziel - Routing in TCP/IP-Netzwerken; UNIX Magazin Ausgabe 8 / August 1992

[7] P. Merdian: Rechnernetze; Regionales Rechenzentrum der Universität Stuttgart

[8] P. Schicker: Datenübertragung und Rechnernetze; B.G. Teubner, Stuttgart

[9] P. Borowka: Brücken und Router - Wege zum strukturierten Netzwerk; Datacom

[10] A. Dripke: Netzwerke im Überblick - Grundlagen und Einsatzmöglichkeiten; Verlag: McGraw-Hill

[11] Arnold: Heterogene Netzwerke; Franzis'

[12] Cisco Internetworking Technology Overview

[13] Andrew S. Tanenbaum: Computernetzwerke; Prentice Hall, 3. Auflage 1997

[14] K. Epple: Kein Erfolg ohne Manager; DECKBLATT Ausgabe 1 / Januar 1992

[15] Netz- und Systemmanagement; it + ti Ausgabe 6 / 1996

[16] B. Hackler, P. Mandl: Management von verteilten Anwendungen über Gateway-Mechanismen; PIK Ausgabe 2 / 1995

[17] F.-J. Kauffels: Netzwerk- und System-Management; Datacom, 1995

[18] H.-G. Hegering, s. Abeck: Integriertes Netz- und Systemmanagement; Addison-Wesley (Deutschland), 1. Auflage 1993

[19] S. Abeck: Integriertes Management am Beispiel einer unternehmensspezifischen kommunikationsnahen Anwendung; PIK Ausgabe 4 / 1994

[20] G. Siegmund: Technik der Netze; R.v.Decker, 3. Auflage 1996

[21] G. Rößler, W. Schollenberger: Netzmanagement in heterogenen lokalen Netzen; PIK Ausgabe 4 / 1991

[22] Douglas Comer: Internetworking with TCP/IP, Volume I; Prentice Hall 1995, 3rd Edition

[23] Santifaller, Michael: TCP/IP und NFS in Theorie und Praxis; Addison-Wesley, 1990

[24] Glaser, Gerhard M.: TCP/IP - Protokolle, Projektplanung, Realisierung, Pullheim; DATACOM-Buchverlag, 1990

[25] Dittler, Hans Peter: IPv6 – das neue Internet Protokoll; dpunkt.verlag, 1998

[26] Rose, Marshall T.: Einführung in die Verwaltung von TCP-IP-Netzen; Hanser, 1993