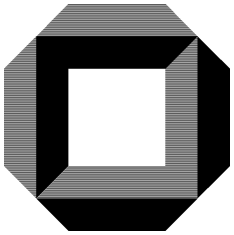


Universität Karlsruhe

Fakultät für Informatik

Informatik-Rechnerabteilung
Institut für Betriebs- und Dialogsysteme

Seminar Netzwerkmanagement

Wintersemester 1994/95

Betreuer und Herausgeber:

Dipl.-Inform. G. Schreiner

Prof. Dr. W. Zorn

Einleitung

Die Datenkommunikation ist in den letzten Jahren wesentlicher funktionaler Bestandteil einer Datenverarbeitungsanlage geworden, so daß fast jedes Rechensystemen in ein Netzwerk eingebunden wird. Während die theoretischen Grundlagen einer DV-Anlage in Bezug auf ihr Inneres in Forschungsbereichen wie Betriebssysteme u.ä. ergiebig untersucht wurde, blieb der Bereich der Netzverwaltung längere Zeit unbetrachtet.

Betrachtet man das Themengebiet "Netzwerkmanagement", lassen sich zwei fast gegensätzliche Entwicklungstendenzen entdecken: zum einen der OSI-Standard, der theoretisch sehr fundiert das Aufgabengebiet angeht, aber aufgrund der Komplexität noch wenig Verbreitung gefunden hat und zum anderen der Standard der TCP/IP-Welt, das Simple Network Management Protocol, welcher ein weites Einsatzfeld vorweisen kann, aber auch noch konzeptionelle Schwachstellen in sich birgt.

Nach einer allgemeinen Einführung in den Bereich Netzwerkmanagement werden die in den zwei führenden Protokollwelten, IP bzw. OSI, standardisierten und im Einsatz befindlichen Konzepte vorgestellt, welche vornehmlich auf einem zentralistischen Management basieren. In den nachfolgenden Vorträgen werden innovative Architekturen erörtert, die den zentralistischen Ansatz insbesondere in Netzen mit hoher Endknotenzahl ergänzen beziehungsweise verdrängen werden.

Der Herausgeber

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einführung in das den Begriff Netzwerkmanagement und Betrachtung des Bereiches Netzwerkkontrolle

Simone Welle

1.1 Einleitung

Netzwerken und verteilten Systemen kommt eine immer größer werdende Bedeutung zu, werden von der Geschäftswelt aber auch kritisch betrachtet. Innerhalb einer gegebenen Organisation geht der Trend hin zu größeren, komplexeren Netzwerken, die immer mehr Anwendungen und Benutzer unterstutzen. In dem Maße, wie die Netzwerke sich vergrößerten, wurden zwei Punkte extrem bedeutend:

1. Das Netzwerk und die damit verbundenen Ressourcen und verteilten Anwendungen wurden für die Organisationen unentbehrlich.

2. Durch entsprechende technische Fehler oder Anwendungsfehler besteht die Möglichkeit, daß das gesamte Netzwerk oder ein Teil davon zusammenbricht oder der Durchsatz auf einen nicht mehr akzeptablen Level herabgesetzt wird.

Ein komplexes Netzwerk einzurichten und zu verwalten, ist ohne technische Hilfsmittel kaum möglich. Als Hilfsmittel dienen automatische Netzwerkmanagement-Tools, die aus Kostengründen in standardisierter Form angeboten werden und für ein breites Spektrum von Produkttypen verwendet werden können (z.B. Endsysteme, Bridges, Routers, Telekommunikationseinrichtungen,...). Ausgehend von dieser Notwendigkeit entwickelten sich zwei standardisierte Arbeiten, nämlich die SNMP-Familie und das OSI-System-Management, die in späteren Vorträgen beschrieben werden.

1.2 Definition eines Datennetzwerks

- Ansammlung von Geräten und Verbindungsleitungen, die dazu dienen, Daten von einem Rechner zum anderen zu transferieren.

- Mehrere Benutzer an unterschiedlichen Orten können auf die Ressourcen eines entfernten Hauptrechners mit hoher Rechnerleistung zugreifen, sodaß eine effiziente und produktive Nutzung möglich ist, was zeit- und kostensparende Vorteile mit sich bringt.

1.3 Implementierung eines Datennetzwerks

Um ein Datennetzwerk zu implementieren, muß zuerst ein Plan erstellt werden, der den Ansprüchen der Benutzer gerecht wird. Nachdem dieser Plan entwickelt wurde, sollte der Netzwerkingenieur für die Implementierung des Netzes folgende Aufgaben durchführen:

- aufbauen
- warten
- erweitern
- optimieren
- kontrollieren

1. Aufbau:

   Feststellen, welche Hard- und Softwarekomponenten benötigt werden und welches die gewünschten Verbindungen sind. Es gibt zwei Haupttypen von technologischen Verbindungen zwischen zwei miteinander kommunizierenden Punkten:

   - Lokales Netzwerk (LAN: local area network): Dieses überträgt die Daten zwischen den Host-Rechnern mit einer Übertragungsgeschwindigkeit von 4 - 400 Mbps (megabits per second) und wird für relativ kurze Entfernungen eingesetzt.

- Globales Netzwerk (WAN: wide area network): Das WAN hat eine Datenübertragungsrate von 9.6 Kbps bis 45 Mbps und dient einer Datenübertragung über relativ weite Distanzen.

2. Wartung:

   Nachdem das Netz aufgebaut ist, muß es beispielsweise aufgrund neuer Upgrades für Teile des Netzwerks oder wegen fehlerhafter Teile, die ersetzt werden müssen, ständig gewartet werden.

3. Erweiterung:

   Es sollte die Möglichkeit einer einfachen Erweiterung des Netzwerks gegeben sein, ohne ein komplett neues entwerfen zu müssen, d.h. der Netzwerkingenieur muß bereits bei der Implementierung eine später mögliche Erweiterung vorsehen, sodaß er dann auch tatsächlich einfache und korrekte Netzwerk-Erweiterungen anbieten kann.

4. Optimierung:

   Das Datennetzwerk besteht üblicherweise aus mehreren hundert unterschiedlichen Geräten, jedes mit seinen eigenen Besonderheiten. Diese Geräte müssen miteinander, ohne Probleme zu verursachen, kommunizieren können. Die Aufgabe des Netzwerk-Ingenieurs ist es, ein solches Datennetzwerk so zu optimieren, daß es die maximale Leistung erbringt.

5. Kontrolle:

   Durch die oben genannten Punkte kann der Netzwerk-Ingenieur zwar eventuell auftretende Probleme minimieren, aber nie vollständig vermeiden, was zu diesem letzten Punkt führt: nämlich Probleme, die durch unvorhergesehene Ereignisse auftreten können, festzustellen.

1.4 Netzwerkmanagement

Netzwerkmanagement bedeutet, ein komplexes Netzwerk so zu verwalten, daß dessen Effizienz und Produktivität maximal wird. Um dieses Ziel zu erreichen, sind fünf funktionale Bereiche zu nennen, aus dem das Netzwerk-Management besteht:

- Fault Management
- Configuration Management
- Security Management
- Performance Management
- Accounting Management

Diese fünf Bereiche wurden von dem International Organization for Standards (ISO) Network Management forum definiert und werden nun in den folgenden Abschnitten näher erläutert.

1.4.1 Fault Management

Um einen einwandfreien, ununterbrochenen Netzwerk-Betrieb gewährleisten zu können, muß auf die korrekte Funktionalität des Netzes als Ganzes, aber auch jeder einzelnen Komponente geachtet werden. Die Dienste des Fault Managements umfassen Tests zur Überprüfung von Netzkomponenten bzw. zur Fehlerlokalisierung, es können aber auch Fehlermeldungen ausgetauscht oder Fehlerstatistiken abgerufen werden.

Tritt ein Fehler auf, so ist es wichtig, folgende Schritte möglichst schnell auszuführen:

- Exakt feststellen, woran der Fehler liegt.
- Isolation des restlichen Netzwerks von der Störungsstelle, sodaß es ohne Unterbrechung weiterverwendet werden kann.
- Modifikation oder Rekonfiguration des Netzwerks, um eine Verwendung der fehlerhaften Komponente(n) zu minimieren.
- Reparatur oder Erneuerung der fehlerhaften Komponenten, um das Netzwerk wieder in seinen ursprünglichen Zustand zu bringen.

Benutzer erwarten schnelle und zuverlässige Problemlösungen. Durch die Verwendung von schnellen und zuverlässigen Fehlerverwaltungstechniken, z.B. dem Fault Management Tool, hat der Netzwerk-Ingenieur die Möglichkeit, auftretende Probleme wesentlich schneller zu lokalisieren und zu lösen als ohne diese. Die Wirkung und Dauer von Fehlern können aber auch durch die Verwendung von redundanten Komponenten und alternierenden Kommunikationsrouten minimiert werden und dem Netzwerk dadurch eine gewisse Fehlertoleranz gewähren.

Wenn ein Fehler korrigiert wurde und das System sich wieder in seinem ursprünglichen Zustand befindet, muß der Fault-Management-Dienst sicherstellen, daß das Problem auch tatsächlich beseitigt wurde und dabei keine neuen Probleme entstanden sind.

1.4.2 Configuration Management

Das Configuration Management bietet dem Netzmanager die Möglichkeit, Kontrolle über das System bzw. Netz auszuüben. Hierzu stehen Funktionen zur Verfügung, um Parameter, die den normalen Betrieb des Netzes beeinflussen, zu modifizieren, aber auch, um die Konfiguration des Netzes zu modifizieren. Die Konfiguration von bestimmten Netzwerkgeräten bestimmt das Verhalten des Datennetzwerks. Dieses besteht i.a. aus individuellen Komponenten und logischen Subsystemen, die verschiedene Anwendungen durchführen können. Dasselbe Gerät kann beispielsweise so konfiguriert sein, daß es entweder als Router, als Endsystem oder als beides fungieren kann. Ist die gewünschte Funktionalität eines Gerätes bestimmt, so kann der Konfigurationsmanager die geeignete Software auswählen und die Geräte entsprechend konfigurieren.

Falls es ein Upgrade einer Software-Version gibt, muß dieses auf sämtlichen Geräten installiert werden. Durch ein Configuration-Management-Tool wird dem Ingenieur eine Liste der aktuellen Software-Versionen sämtlicher Geräte geliefert, sodaß dieser nicht jedes einzelne Gerät überprüfen muß. Benutzer wollen bzw. benötigen den jeweils aktuellen Status der Netzwerk-Ressourcen und -Komponenten. Diese Informationen erfolgen in Form von aktuellen Konfigurations-Berichten - entweder periodisch oder auf Anfrage der Benutzer.

1.4.3 Security Management

Als Security Management wird der Prozeß bezeichnet, der die Zugriffsberechtigung auf Netzwerkdaten kontrolliert und schützt. Dies ist damit verbunden, daß Kodierschlüssel generiert, verteilt und gespeichert werden. Dadurch kann nur ein autorisierter Benutzer auf die Daten zugreifen. Im Falle eines unerlaubten Zugriffs besteht durch das Security Management die Möglichkeit, diese Zugriffe auf den Terminal-Server aufzuzeichnen und dadurch u.U. die entsprechende Person festzustellen. Weiter gibt es auch die Möglichkeit, daß bei unbefugtem Zugriff Alarme ausgelöst werden.

1.4.4 Performance Management

Das Performance Management überwacht die Leistungsfähigkeit der einzelnen Systeme und des gesamten Netzes. Moderne Datenkommunikationsnetzwerke bestehen aus vielen und unterschiedlichen Komponenten, die miteinander kommunizieren und Daten und Ressourcen teilen müssen. In manchen Fällen ist es problematisch, daß sich die Kommunikation über das Netzwerk an einer bestimmten Durchsatzgrenze befindet, wodurch Engpässe entstehen können. Solche Engpässe müssen frühzeitig festgestellt und beseitigt werden können. Aus diesem Grunde und um die Leistungsfähigkeit zu verbessern, enthält das Performance Management Funktionen, die die statistischen Informationen bzgl. der Leistungsfähigkeit des Systems abrufen und die Konfiguration von Netzkomponenten modifizieren. Bevor ein Benutzer ein Netzwerk für eine bestimmte Anwendung einsetzt, möchte er genaue Informationen über die durchschnittliche und schlechteste Antwortzeit und die Zuverlässigkeit von Netzwerkdiensten. D.h., der Durchsatz muß in ausreichendem Maße bekannt sein, um auf spezielle Benutzerfragen antworten zu können. Endbenutzer erwarten, daß die Netzwerkdienste auf eine solche Art und Weise durchgeführt werden, daß ihre guten Antwortzeiten dauerhaft gewährleistet sind. Netzwerkmanager benötigen Durchsatzstatistiken zur Planung und Verwaltung von großen Netzwerken. Bezüglich den Endbenutzern dienen die Statistiken auch dazu, potentielle Engpässe zu entdecken und geeignete Verbesserungsmaßnahmen anzuwenden, noch bevor sie den Endbenutzern Probleme bereiten.

1.4.5 Accounting Management

Das Accounting Management unterstützt bei Erreichen eines Limits das Aushandeln von benötigten Ressourcen einer Gruppe oder eines Benutzers. Die Zuteilung erfolgt entweder durch die Vergabe von Prioritäten oder durch den Einsatz zusätzlicher File-Server. Weiter ist das Accounting Management für die Gebührenverwaltung und ähnliche Aufgaben zuständig: Um die Kosten, die durch ein Netzwerk entstehen, zumindest zu decken, muß dessen Verwendung irgendwie berechnet und bezahlt werden. Dies muß nicht unbedingt durch reales Geld, sondern kann auch durch eigene, zur Verfügung gestellte Dienstleistungen erfolgen. Auch wenn solch eine Vergütung nicht eingesetzt wird, so muß der Netzwerkmanager doch feststellen können, welcher Benutzer oder welche Benutzergruppe das Netzwerk verwendet. Gründe:

- Ein Benutzer oder eine Benutzergruppe mißbraucht ihre Zugriffsprivilegien und belastet das Netzwerk unnötig auf Kosten der anderen Benutzer.
- Benutzer können das Netzwerk ineffizient verwenden, sodaß der Netzwerkmanager sie dabei unterstützen kann, einzelne Prozeduren zu verändern, um die Leistung zu verbessern.
- Der Netzwerkmanager kann die Erweiterung des Netzwerks besser planen, falls er die Benutzer-Aktivitäten in ausreichendem Maße kennt.

1.5 Netzwerk-Management-Systeme

Netzwerk-Management-Systeme sind eine Ansammlung von Geräten/Werkzeugen zur Netzwerküberwachung und -kontrolle, was bedeutet:

- Eine einzelne Operator-Schnittstelle mit einer mächtigen, aber benutzerfreundlichen Menge von Befehlen, um die meisten Netzwerk-Management-Aufgaben durchzuführen.
- Eine minimale Anzahl von getrennten Geräten. D.h., die meiste Hard- und Software, die für das Netzwerk-Management erforderlich ist, wird mit der jeweils existierenden Benutzerausstattung implementiert.

Die Software, die für die Ausführung der Netzwerk-Management-Aufgaben eingesetzt wird, ist in den Host-Rechnern und Kommunikationsprozessoren (z.B. Front-End-Prozessoren, Bridges und Routers) implementiert. Ein Netzwerk-Management-System wird unter dem Gesichtspunkt entwickelt, das ganze Netzwerk als eine vereinheitlichte Architektur zu betrachten, wobei jeder Punkt durch Adressen und Labels gekennzeichnet ist, und jedes spezifische Attribut eines Elements und jede

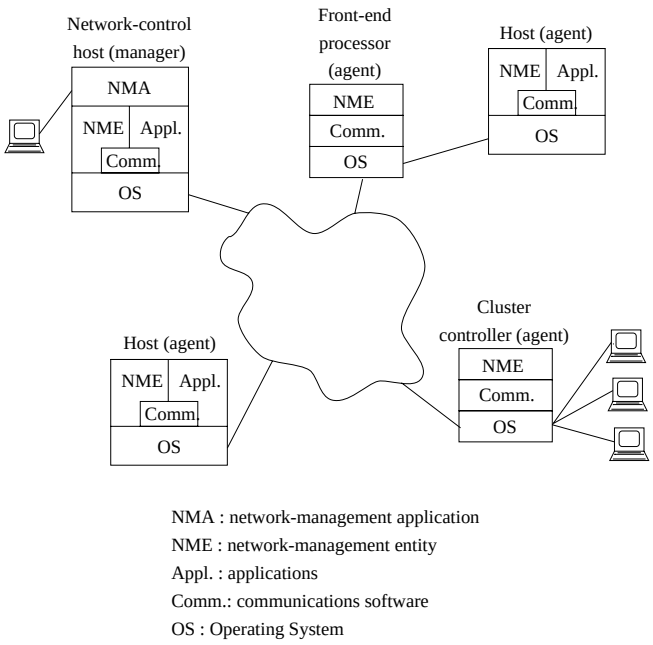


Abbildung 1.1: Elemente eines Netzwerkmanagement-Systems

Verbindung dem System bekannt ist. Die aktiven Elemente eines Netzwerks liefern an das Netzwerk-Kontroll-Zentrum eine regelmäßige Rückmeldung über die Zustandsinformationen.

Abbildung 1.1 zeigt eine mögliche Architektur eines Netzwerk-Management-Systems. Jeder Netzwerk-Knoten besitzt eine Reihe von Software, im Bild als eine Netzwerk-Management-Einheit (NME) dargestellt. Jede NME führt folgende Aufgaben durch:

- Sammlung von Statistiken über die Kommunikationen und netzwerkbezogenen Aktivitäten.
- Lokale Speicherung dieser Statistiken
- Antworten auf Anweisungen des Netzwerk-Kontroll-Zentrums. Anweisungen können sein:
  - Übergabe der Statistiken an das Netzwerk-Kontroll-Zentrum
  - Änderung eines Parameters (z.B. ein Zeitgeber (timer) in einem Transport-Protokoll)
  - Liefern von Status-Informationen (z.B. Parameterangaben, aktive Verbindungen)
  - Generieren von Verbindungen für Testzwecke

Mindestens ein Host im Netzwerk wird als Netzwerk-Kontroll-Host (Manager) bestimmt. Dieser besitzt neben der NME-Software noch eine weitere Software-Ansammlung, die auch als Netzwerk-Management-Anwendung (NMA) bezeichnet wird. Diese NMA beinhaltet eine Operator-Schnittstelle, um einem autorisierten Benutzer die Verwaltung des Netzwerks zu erlauben. Die NMA liefert Antworten auf Benutzeranweisungen, indem sie Informationen aufzeigt und/oder Befehle über das Netzwerk an die NMEs weiterleitet. Diese Verbindung wird ausgeführt unter Verwendung eines Anwendungsschicht-Netzwerk-Protokolls, das die Kommunikations-Architektur genauso verwendet wie jede andere verteilte Anwendung auch.

Jeder andere Knoten im Netzwerk, der Teil des Netzwerk-Management-Systems ist, besitzt ebenfalls eine NME und wird für Zwecke des Netzwerk-Managements als Agent bezeichnet. Agenten sind End-Systeme, die Benutzeranwendungen unterstützen ebenso wie Knoten, die Kommunikationsdienste anbieten, wie ein Front-End-Prozessor, Cluster-Controller, Bridges und Routers.

1.6 Literaturverzeichnis

[LF89] [Sta93f] [Sta93b]

Kapitel 2

SNMPv2: Konzept und Managementdatenbasis

Armin Schäfer

2.1 Einführung

2.1.1 Entstehungsgeschichte

Die Entwicklung von SNMP *(Simple Network Management Protocol)* folgt einem historischen Muster, ähnlich wie die Entwicklung der TCP/IP *(Transmission Control Protocol / Internet Protocol)* Protokollfamilie.

Diese begann im Jahr 1969, als die DoD *(U.S. Department of Defense)* das erste paketvermittelte Netzwerk "ARPANET" entwickelte. Mit der stark wachsenden Anzahl an zu verbindenden Hosts und Terminals von verschiedensten Herstellern, sowie der Entfaltung des ARPANET zum Kern eines Internet, war man sich über die Notwendigkeit eines standardisierten Protokolls einig, um die Betriebsfähigkeit eines solch heterogenen Netzes zu gewährleisten.

Somit wurden Ende der Siebziger der Grundstein für die heutige TCP/IP Protokollfamilie gelegt. Die Nachfrage nach TCP/IP stieg auch im nichtmilitärischen Bereich permanent an.

Zur gleichen Zeit etwa begann die OSI– *(Open Systems Interconnection)* Entwicklung, die sich jedoch gegenüber TCP/IP aufgrund höherer Komplexität und Nichtvorhandensein einer lauffähigen Version nicht durchsetzen konnte.

Die IAB *(Internet Activity Board)* legte 1988 einen geteilten Ansatz in der Entwicklungsstrategie fest:

Kurzfristig sollte das bereits bestehende, lauffähige Verwaltungsprotokoll SGMP *(Simple Gateway Monitoring Protocol)* mit Hilfe der gesammelten praktischen Erfahrungen zu SNMP weiterentwickelt werden.

Langfristig baute die IAB auf die Entwicklung des OSI-Protokolls CMIP *(Common Management Information Protocol)* auf TCP/IP Basis bzw. CMOT *(CMIP over TCP/IP)*.

Die IAB legte mit der SMI *(Structure of Management Information)* und der MIB *(Management Information Base)* die gemeinsame Nutzung einer Objektdatenbank fest, um den späteren Umstieg auf OSI zu erleichtern.

Dieser Ansatz stellte sich jedoch sehr bald als ziemlich inpraktikabel heraus, woraufhin die IAB die Bestimmung einer gemeinsamen SMI und MIB wieder aufgab und zu einer parallelen Entwicklungsstrategie von SNMP und CMOT überging.

Auf der "SNMP–Seite" kam es zu schnellen praktischen Entwicklungen durch einen einfachen Aufbau und den exakten Protokolldefinitionen, wogegen es auf der "OSI–Seite" nur zu schleppenden, inpraktikablen Entwicklungen kam.

Heute hat sich SNMP zu einem De–facto–Standard entwickelt und wird von allen führenden Herstellern aus dem Netzwerkbereich unterstützt. Im April 1993 wurde es durch eine noch leistungsfähigere und umfassend überarbeitete SNMP–Spezifikation "SNMPv2" ersetzt. SNMPv2 wird den Anforderungen an ein modernes Managementprotokoll gerecht.

2.1.2 Was leistet SNMP ?

Das SNMP–Protokoll wird als Mechanismus zur Bereitstellung und zum Transport von Managementinformationen zwischen den Komponenten am Netzwerk eingesetzt. Es ermöglicht einem Netzwerkadministrator die interaktive Abfrage von Parametern oder die berwachung von bestimmten Netzwerkzuständen. Allgemein ausgedrückt ermöglicht das SNMP-Protokoll das Managen aller SNMP–Geräte am Netz. Auf der Basis des SNMP–Protokolls werden alle für die Managementapplikationen notwendigen Daten (wie z.B. Status, Performance, Fehler, Alarme, Reports usw.) zwischen den managebaren Geräten übermittelt.

2.1.3 Standards, die SNMP beschreiben

Es gibt drei grundlegende Spezifikationen, die das SNMP–Protokoll, die zugehörigen Funktionen und die Datenbank beschreiben. Eine solche Spezifikation heißt RFC *(Request For Comment)* und wird mit einer fortlaufenden Nummer eindeutig gekennzeichnet.

SNMP wird durch folgende RFC's beschrieben:

1. *Structure and Identification of Management Information for TCP/IP-based Internets* (RFC 1157) **(SMI)**
   ⇒ beschreibt, wie managebare Objekte in der MIB definiert werden;
2. *Management Information Base for Network Management of TCP/IP-based Internets* (RFC 1213) **(MIB)**
   ⇒ beschreibt, die in der MIB enthaltenen managebaren Objekte;
3. *Simple Network Management Protocol* (RFC 1157) **(SNMP)**
   ⇒ definiert das Potokoll, um die Objekte zu managen.

Bemerkung

Die RFCs werden in regelmäßigen Abständen von dem höchsten Standardisierungsgremium IAB "upgedated", so daß die o.a. RFC–Nummern nur beispielhaften Charakter haben.

2.1.4 SNMP Netzwerkmanagement Architektur

Die Netzwerkmanagement–(NM–) Architektur besteht aus den folgenden Schlüsselelementen:

- Network Management Station **(NMS)**
- Network Management Agent **(NMA)**
- Management Information Base **(MIB)**
- Network Management Protocol **(SNMP)**
- Community

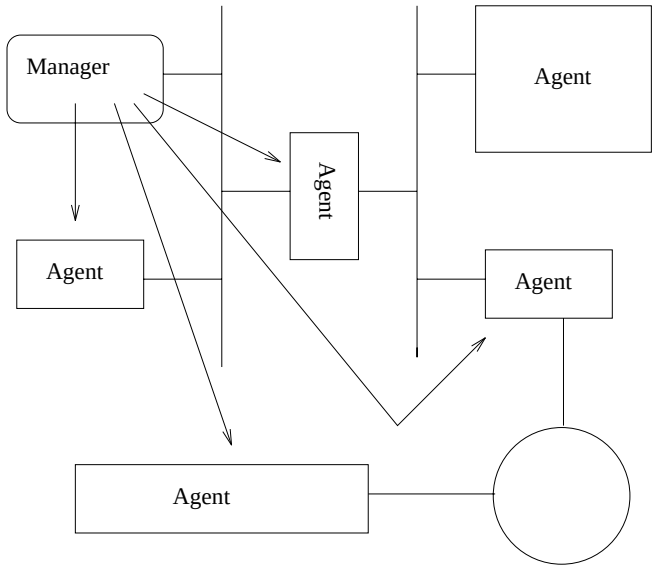


Abbildung 2.1: Manager-Agent-Beziehung

Die NM–Architektur basiert auf einem Modell, das aus einer oder mehreren NMS's und mehreren managebaren Netz-Objekten besteht. Eine **NMS** überwacht und kontrolliert bestimmte Netz-Objekte. Netz-Objekte werden als Geräte im Netz definiert (z. B. Host, Bridge, Router etc.), die einen oder mehrere Agents (Software Modul) implementiert haben. Auf dem **Agent** werden bestimmte NM-Tasks ausgelagert und die Verarbeitung findet vor Ort im Netz-Objekt statt. Auf einer NMS laufen neben dem Agent NM-Applikationen. Das **SNMP** Protokoll dient der Kommunikation zwischen dem Agent und der NMS. Alle managebaren Ressourcen im Netz werden in Form von Objekten repräsentiert. Die strukturierte Ansammlung dieser Objekte in einer Datenbank stellt die **MIB** dar. Die Kommunikationsbeziehungen zwischen NMS und Agent werden in Gruppen eingeteilt. Miteinander kommunizierende Gruppen werden als **Community** bezeichnet. Jede Community besitzt einen Namen *(Community String)* zur Identifikation. Durch die genau festgelegten administrativen Beziehungen zwischen NMS und Agent erweist sich dieses Modell als eine relativ starre Architektur.

Man bezeichnet dieses Modell als

"Client–Server–Modell"
("Query–Response–Modell")

was folgendermaßen zu verstehen ist:

> Der **Client**, der die Anfragen (Queries) aussendet wird allgemein als der "Manager" bezeichnet.
>
> Der **SNMP–Server**, das Gerät, das auf die Queries antwortet, ist als "Agent" bekannt.
>
> Und das SNMP–Protokoll ermöglicht einer NMS, nach den Regeln des SNMP, die Parameter eines Agents zu lesen und zu modifizieren.

2.2 Managementdaten

2.2.1 Was ist eine MIB ?

Ein auf TCP/IP basierendes Netzwerk Management System besitzt als Grundlage eine Datenbank, die sämtliche Informationen über die zu managenden Objekte (Netz-Ressourcen) enthält.

In der TCP/IP Welt (auch in einer OSI-Umgebung) wird eine solche Datenbank als **Management Information Base (MIB)** bezeichnet. Jede Ressource, die einen bestimmten Teilbereich des Netzes darstellt, ist durch ein Objekt in der MIB repräsentiert, d.h. die Statuswerte dieser Ressource sind in der MIB "abgelegt" und können z.B. von einer NMS abgefragt oder modifiziert werden.

Damit die MIB den Ansprüchen des NM-Systems gerecht werden kann, müssen zwei Richtlinien eingehalten werden:

1. Jede Ressource in einem Netzwerk muß mit den gleichen Objekten beschrieben werden (z. B. offene Verbindungen über einen bestimmten Zeitraum)

→ wird durch Objektdefinition und MIB-Struktur festgelegt

2. Verwendung eines gemeinsamen Schemas zur Repräsentation, um die Interoperabilität zu unterstützen
   → wird durch die SMI festgelegt

2.2.2 Struktur der Managementdaten (SMI)

Die Aufgabe der SMI ist es, einen Standard für die Repräsentation von Management Informationen zu definieren, d.h. sie legt einen "generellen Rahmen" fest, innerhalb dessen eine Management Information Base (MIB) definiert und konstruiert werden kann.

Die Philosophie, die hinter der SMI steckt, ist die Einfachheit und Erweiterbarkeit innerhalb der MIB zu garantieren. Aus diesem Grund kann die MIB auch nur einfache Datentypen aufnehmen, aus denen sich jedoch komplexere Datenstrukturen konstruieren lassen. Komplexe Datenstrukturen werden deshalb von der SMI vermieden, um die Implementation zu simplifizieren und die Interoperabilität zu erweitern.

Um jedoch eine standardisierte Repräsentation von Managementdaten zu erhalten, müssen entsprechende standardisierte Definitionsmechanismen zur Verfügung stehen:

- Definition der MIB-Struktur
- Definition der Objekte, sowie deren Syntax und Value
- Kodierungsregeln

Zur Darstellung aller Informationen und Objekten wird eine formale Spezifikationssprache **Abstract Syntax Notation One (ASN. 1)**[1] verwendet. Durch ihre Verwendung erreicht man eine präzise und unzweideutige Definition der MIB.

Struktur der MIB

Alle Managementinformationen werden als Objekte (ObjectType) dargestellt, die in einer oder mehreren MIBs zusammengefaßt und durch die SMI festgelegt werden. Die SMI legt fest, daß die zu managenden Informationen und Daten eindeutig zu identifizieren sein müssen. Aus diesem Grund wurde für die Darstellung der Daten in der MIB eine baumartige Struktur verwendet (aus der OSI–Definition übernommen).

Dieser Managementbaum ist von der Wurzel aus streng hierarchisch aufgebaut. Der Weg durch diesen Baum, hin zu einem Knoten oder Blatt, wird als Objekt–Identifikator (OBJECT IDENTIFIER)[2] gekennzeichnet. Dem Baum ist bezüglich seiner Tiefe keine Grenze vorgegeben. Jeder Knoten außer der Wurzel ist mit einem Namen und einer Zahl zur Identifikation versehen.

Das Bild 2.2 zeigt den für das Netzwerkmanagement wichtigen Ausschnitt des Managementbaums.

[1] mehr über ASN.1 in 2.5
[2] vgl. ASN.1 in 2.5

Der Suchbaum der ISO, mit dem Label `iso(1)`, ist der erste Nachfolger der Wurzel des Managementbaums. Unterhalb des iso(1) wurde für andere nationale und internationale Standardisierungsorganisationen mit dem Label `org(3)` ein weiterer Suchbaum eingerichtet. Dieser Suchbaum wurde der U.S. National Institutes of Standards and Technology (NIST) zur freien Verfügung gestellt, wobei einer dieser "NIST-Suchbäume" der DoD überlassen wurde — mit dem Label `dod(6)`. Von der Internet–Community wurde einer der "DoD–Suchbäume" mit dem Label **internet(1)** belegt.

Unterhalb des "Internet–Suchbaums" sind alle für das Internet wichtigen Management Suchbäume angeordnet. Diese sind u.a.:

`directory(1):` ist für zukünftige Anwendungen reserviert ,

`mgmt(2):` ist in jeder SNMP Implementierung zwingend vorgeschrieben,

`experimental(3):` experimenteller Subzweig; wird nur bei Versuchen im Internet (Test neuer Objekte) benutzt,

`private(4):` über die Internet Assigned Numbers Authority kann sich jeder Hersteller eine eigene Herstellerkennung registrieren lassen und unter dieser Kennung seine eigenen privaten Objekte definieren
⇒ Hilfsmittel zur Integration herstellerspezifischer Objekte, die über die standardmäßig festgelegten Objekte hinausgehen.

Unumgänglich bei einer SNMP–Implementierung ist der Suchbaum `mgmt(2)` mit einem seiner Suchbäume, dem "MIB–II–Suchbaum" mit dem Label `mib-2(1)`[3]. Unterhalb der "MIB–II–Suchbaums" sind die Objektgruppen der MIB–II angeordnet.

Beispiel Der Knoten `mib-2(1)` stellt die Wurzel aller SNMP verwalteten Daten dar. Um diesen Knoten unter ASN.1 anzusprechen muß folgender Objekt–Identifikator

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1)
```

(oder Integersequenz `1.3.6.1.2.1`) benutzt werden. Diese Folge ist Präfix für alle Variablen in der MIB–II.

Objektgruppen der MIB

Die MIB beschreibt bekanntlich die Objekte, die von verwalteten Netzknoten unterstützt werden sollen und auf denen die Internet Protokollfamilie implementiert ist. Also müssen alle Geräte, die verwaltete Knoten sind, die MIB implementieren. Da es jedoch nicht notwendig ist, alle Funktionen auf allen Knoten zu implementieren (z.B. wird ein Gateway keine Verwaltungsobjekte für elektronische Post unterstützen), sind die Objekte der MIB in Gruppen (Objektgruppen) eingeteilt.

[3] die MIB–I (1988) wurde 1990 durch die MIB–II ersetzt

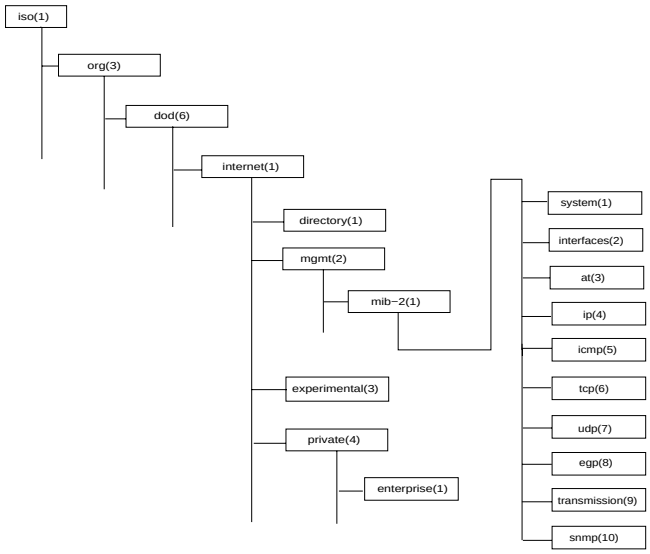


Abbildung 2.2: Managementbaum

Nr.	Objektgruppe	enthält Informationen über
1	`system`	enthält Informationen über das managed object aus administrativer Sicht
2	`interface` (`if`)	enthält die Parameter aller im Gerät installierten Netz–Schnittstellen
3	address translation (`at`)	dient zur Umsetzung von medienspezifischen Adressen in Netzadressen (z.B. Ethernet– in IP–Adressen) und umgekehrt
4	internet protocol (`ip`)	enthält mehrere Zähler und Tabellen, die die IP–Aktivität im Gerät insgesamt beschreiben
5	internet control protocol (`icmp`)	enthält mehrere Zähler, die sich auf die Anzahl von gesendeten und empfangenen ICMP–Nachrichten beziehen
6	transmission control protocol **(tcp)**	enthält mehrere Statistik–Zähler zu TCP–Verbindungen und gesendeten bzw. empfangenen TCP–Packeten, sowie eine Tabelle der TCP–Verbindungen
7	user datagram protocol **(udp)**	enthält Informationen über den Datagrammtransport
8	exterior gateway protocol **(egp)**	enthält Informationen über die Erreichbarkeit zwischen unabhängigen Systemen
9	**transmission**	Platzhalter für herstellerabhängige MIBs
10	simple network protocol **(snmp)**	enthält alle Objekte, die SNMP–Anwendungseinheiten beschreiben

Tabelle 2.1: Objektgruppen

In der MIB–I (1988) gab es ursprünglich acht Objektgruppen (1–8), die dann 1990 mit der MIB–II um zwei Objektgruppen (9–10) erweitert wurde (siehe Tabelle 2.1).

Objekt Syntax

In der SMI werden nicht die einzelnen managebaren Objekte als solche, sondern ihr formaler Aufbau und ihre Inhalte mittels ASN.1 definiert. Jedes Objekt (ObjectType) besteht aus fünf Feldern:

- `ObjectName`: Name des Objekts
- `SYNTAX`: definiert den Datentyp
- `DEFINITION`: textuelle Beschreibung
- `ACCESS`: definiert den erlaubten Zugriff
- `STATUS`: definiert die Anforderungen an die Implementierungen

Nach folgendem kleinen Beispiel wird ein managebares Objekt namens `sysDescr` (*system description*) definiert[4]:

```
sysDescr OBJECT–TYPE
SYNTAX OCTET STRING
ACCESS read–only
STATUS mandatory
::= {system 1}
```

Die erste Variable (instanziertes Objekt) in der Systemgruppe hat den Namen {`system 1`} oder `1.3.6.1.2.1.1.1`.

2.3 Kritik an SNMP

SNMP hat sich langsam entwickelt und ist heute in Bereiche vorgestoßen, für die es nie konzipiert war. Von den ersten einfachen Protokollmechanismen des SGMP *(Simple Gateway Monitoring Protocol)* bis hin zu den heute erheblich gestiegenen Ansprüchen und den damit verbundenen Funktionalitäten an das SNMP–Netzwerkmanagement haben sich immer wieder Problembereiche herauskristallisiert, die in den ursprünglichen SNMP–Spezifikationen nicht vorgesehen waren.

So sind durch eine Vielzahl von komplexen Gräten (Gateways, Bridges, Router etc.) und deren detaillierten Informationsfluten erhöhte Anforderungen an die Protokollmechanismen des Netzwerk Management Protokolls entstanden.

2.3.1 Starre Client–Server–Architektur

Aufgrund der fehlenden Manager–to–Manager Kommunikationsmöglichkeit können nur "flache" Netzwerkmanagementstrukturen realisiert werden. Ein Manager muß immer das gesamte Netz oder zumindest die gesamte Community überwachen. Wird ein Netz in mehrere Communities unterteilt und werden darüberhinaus noch mehrere NMSs eingesetzt, so kann es zur Bildung von Daten– und Informationsinseln kommen, die untereinander nicht kommunizieren können.

Außerdem ist es für einen Manager nicht möglich, nur auf bestimmte Datenbereiche, die nach vorgegebenen Kriterien "maskiert" werden, zuzugreifen. Eine NMS hat nur die Möglichkeit entweder auf alle Daten oder auf gar keine Daten zuzugreifen.

2.3.2 TCP/IP Grundlage

Das SNMP kann von der Theorie her grundsätzlich auf jedem verfügbaren Protokollstack aufgesetzt werden. In der Praxis hat sich jedoch gezeigt, daß die internen Strukturen (Adressen, reservierte Ports etc.) des SNMP–Protokolls zu unflexibel sind, um in eine Multiprotokoll–Welt ohne großen Aufwand integriert werden zu können. In der Regel haben sich für die SNMP–Anwendungen die TCP/IP Protokolle durchgesetzt. Nur vereinzelt wurden Implementationen entwickelt, die auf anderen Protokollen (wie z.B. IPX) basieren.

2.3.3 Sicherheitsmängel

Im Bereich der Sicherheit gibt es bei der SNMP–Version 1 u.a. folgende Probleme:

Fehlender Authentifizierungsmechanismus: In einem durch das SNMP–Protokoll überwachten und gesteuerten Netzwerk ist es jederzeit möglich, daß eine nicht autorisierte Person die Datenpakete abfängt und die darin enthaltenen Informationen für seine Zwecke verändert. Nach dieser Manipulation werden die veränderten Datenpakete zur eigentlichen Zielstation weitergesendet. Das Empfängergerät hat keine Möglichkeit, diese Datenmanipulation zu erkennen.

Fehlende Reihenfolgerichtigkeit: In der Regel werden alle Daten zwischen der Managerstation und den angeschlossenen Agents mittels des ungesicherten User–Datagram–Dienstes verschickt. Da das UDP–Protokoll keine Reihenfolgerichtigkeit der Daten garantiert, können die SNMP–Daten durch Manipulation einer nicht autorisierten Person verzögert oder in geänderter Reihenfolge beim Empfänger ankommen. Dadurch haben diese Saboteure die Möglichkeit, die Dateninhalte zu ihren Gunsten zu verändern, ohne daß der Empfänger die Manipulation feststellen kann.

Vorspielung einer Community: Durch eine einfache Neudefinierung des Community–Strings ist ein Besitzer einer NMS jederzeit in der Lage, den Zugriff auf jeden an das Netz angeschlossenen Agents zu erreichen. Durch diese Maskerade

[4] Datentypen vgl. 2.5

täuscht ein nicht autorisierter Benutzer eine autorisierte Identität vor und kann alle Informationen auslesen sowie sämtliche Managementoperationen durchführen. Der Agent verfügt über keinerlei Möglichkeit, zwischen der richtigen und falschen Identität zu unterscheiden.

Mitlesen der Informationen: Das passive Mitlesen von Daten durch nichtautorisierte Personen mittels eines Datenanalysators liegt in der Natur der Datennetze. Alle Daten können im LAN passiv mitgelesen werden und zum geeigneten Zeitpunkt mißbraucht werden.

2.4 Reaktion auf die Kritik: SNMP Version 2

2.4.1 Neues administratives Modell

Mit den Sicherheitsmängeln im SNMP(v1)–Standard befaßte sich die "Secure SNMP-Arbeitsgruppe" (SMP–Gruppe), was schließlich dazu führte, daß im April 1993 nach langen Diskussionen und mühsamen arbeitsreichen Sitzungen die Simple Network Management Protocol Version 2 (SNMPv2)–Spezifikationen in einigen RFCs veröffentlicht wurden.

Die SNMPv2–Spezifikationen legen die Grundlage für die Anforderungen an ein modernes Managementprotokoll. Das neue administrative Modell verfügt über nachfolgende grundlegende Schlüsselelemente.

Party–Konzept

Infolge der Sicherheitsprobleme, die sich im praktischen Betrieb des SNMPv1–Protokolls herauskristallisiert haben, wurde in SNMPv2 dafür gesorgt, daß die SNMP–Pakete nicht von nicht autorisierten Stationen abgefangen und modifiziert werden können. Dazu wurde die gesamte Funktion des Community–Strings zugunsten eines Authentifizierungsmechanismus ausgewechselt, der die gesamten Zugriffsrechte und Konfigurationen über Parties regelt.

Was ist eine Party ? Eine SNMPv2 Party beschreibt eine virtuelle Umgebung in einem SNMPv2–Gerät, innerhalb der ein Kommunikationspartner alle Aktionen ausführen kann. Will ein SNMPv2–Gerät *Source Party* eine Message an einen Kommunikationspartner *Destination Party* schicken, so agiert es immer gemäß der durch die Party–Definition vorgegebenen Restriktion (Operationen, Kommandos etc.).

Jedes SNMPv2–Gerät (Agent) enthält eine oder mehrere Parties, die alle folgende Elemente enthalten:

- *unverwechselbare Party–Identität*
- *eindeutiges Authentifizierungsprotokoll:* zur zuverlässigen Identifizierung der Source Party (sendende Party) bei Kommunikation
- *privacy protocol:* zum Schutz der übertragenen Daten gegenüber "Lauschangriffen"
- *eindeutig zugeordneter MIB–View:* definiert den Teil der lokalen MIB, der für die Party verfügbar ist
- *logische Netzwerk Lokation:* definiert eine Netzadresse, unter der die Party erreichbar ist

Im Juli 1992 wurde durch die **Party MIB** die Möglichkeit geschaffen, spezielle Parameter von managed objects in einer eigenen MIB abzubilden. Somit wird dem Administrator ermöglicht, individuell über frei definierbare MIB–Objekte an die einzelnen Benutzer Zugriffsrechte zu vergeben. Die Party MIB unterteilt sich in folgende Objektgruppen:

Party-Tabelle: definiert für jede Party den Namen, Verschlüsselungsverfahren, zugehörige Parameter, Authentifizierungsverfahren und Transportadresse,

Acl-Tabelle: legt die Zugriffsrechte der einzelnen Parties fest,

View-Tabelle: definiert die Teilbereiche der MIB, auf die die jeweilige Party zugreifen kann,

Context-Tabelle: enthält eine Liste von MIB Views, die mit einer Kennung versehen sind, aus der hervorgeht, ob der entsprechende MIB View zugreifbar sein soll oder nicht.

MIB View

Für jede der Parties wird ein bestimmter Teilbereich des MIB–Baumes konfiguriert, der **MIB View**. Ein MIB View besteht aus einer Menge von View Subtrees.

Ein **View Subtree** ist ein Knoten (Objekt Identifikator) im MIB–Baum und all seine untergeordneten Suchbäume. Es besteht die Möglichkeit, mit sog. **View Masks** die Menge der benötigten oder erlaubten Informationen des Unterbaums zu reduzieren — durch ausfiltern mit Hilfe der "View–Maske".

Ein MIB View definiert die lokalen Objektinstanzen, die der entsprechenden Party bekannt sein sollen.

Context

Allgemein definiert ein SNMPv2–Context eine Ansammlung von managebaren Objektressourcen, auf die ein SNMPv2–Gerät Zugriff hat. Diese Objektressourcen können entweder **lokal** oder auch **remote** realisiert sein.

Bezieht sich ein SNMPv2–Context auf die lokalen Objektressourcen, so spricht man von einem MIB View. Bezieht er sich auf Objektressourcen, die als remote Ressource definiert sind, so stellt dieser Context eine **Proxy Kommunikationsbeziehung**[5] dar, und das Gerät, auf welches die Source Party zugreift, agiert als Proxy Agent.

[5] Proxy Kommunikation siehe nächster Abschnitt

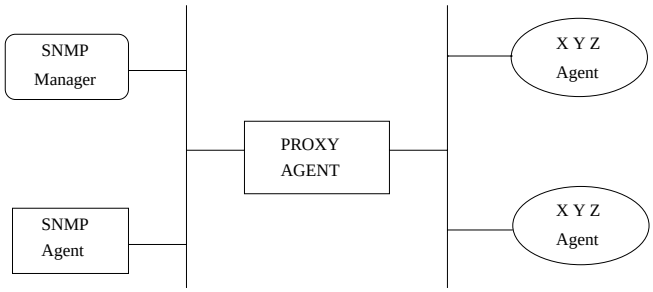


Abbildung 2.3: Proxymanagement

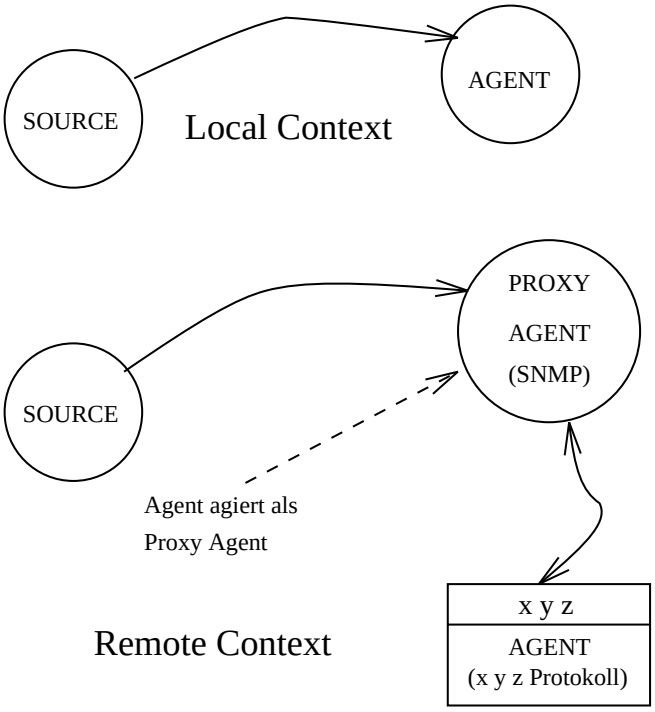


Abbildung 2.4: Local und Remote Context

Proxy Kommunikation

SNMP verlangt, daß alle Agents sowie alle NMSs UDP und IP unterstützen. Diese Einschränkung würde Geräte ausschließen, die nicht auf UDP/IP basieren. Da es jedoch vor der Entwicklung von SNMP zahlreiche andere Entwicklungen gab, die nicht auf UDP/IP aufbauten, wurde das Proxy Konzept entwickelt, um auch solche Geräte, die kein SNMP implementiert haben (wie z. B. Bridges und Router) in das Netzwerkmanagement zu integrieren.

Diese Erweiterung wird als **Proxy Management** bezeichnet. Ein- und zweisprachige Agents ermöglichen bei diesem Ansatz die indirekte Umsetzung der jeweiligen Managementfunktionen.

Access Control Policy

Die Zugriffskontrolle (Access Control) legt die Operationen fest, die eine Source Party bei der Kommunikation mit einer Destination Party anhand des festgelegten Contexts ausführen kann.

Folgende Bereiche definieren die Zugriffskontrolle:

TARGET: definiert die Destination Party, welche die von der Source Party, gewünschten Management Operationen ausführen soll,

SUBJECT: legt die Source Party fest, die in der Lage ist eine gewünschte Management Operation zu veranlassen,

CONTEXT: definiert den MIB View, auf den die Operationen ausgeführt werden sollen,

PRIVILEGES: Zugriffsprivilegien auf Variablen innerhalb eines MIB Views.

2.4.2 Weitere Merkmale der Version 2

In SNMPv2 ist die Möglichkeit der Kommunikation zwischen Managerstationen — *Manager–to–Manager–Kommunikation* — gegeben, d.h. eine NMS kann als Agent oder als Manager agieren (dual role).

Die Einschränkung auf den TCP/IP Transportdienst der SNMP Version 1, ist in der neuen Version aufgelöst. SNMPv2 kann auf *unterschiedlichen Transportdiensten*, wie z. B. DPP (AppleTalk), Novell IPX, OSI etc. betrieben werden.

Mit einer zweisprachigen Realisierung als Überganglösung bis zur vollständigen Migration aller Protokolle zu SNMPv2, wurde das Problem der Rückwärtskompatibilität behoben.

2.4.3 Die SNMPv2 SMI

Gegenüber der Version 1 hat sich die SMI in vielen Bereichen geändert. Im Managementbaum sind unter dem Internet Suchbaum (1.3.6.1) weitere wichtige Verzweigungen angesiedelt worden

- `internet(5): security`

- `internet(6): snmpv2`

Die neuen SNMPv2–MIBs werden unterhalb des SNMPv2 Knoten eingebunden (mit Präfix `1.3.6.1.6`). Die Structure of Management Information (SMI) besteht bei SNMPv2 aus folgenden Teilen:

- Modul-Definitionen
- Objekt-Definitionen
- Tabellen-Definitionen
- Notifikations-Definitionen
- neue Datentypen

Modul–Definitionen: Mit SNMPv2 wird das Konzept der Informationsmodule eingeführt. Ein Modul wie z.B. die MIB spezifiziert eine Gruppe von Objekten.

Objekt–Definitionen: Die einzelnen Objektidentifikatoren werden in der SNMPv2 SMI durch ein OBJECT IDENTITY MACRO festgelegt, welches aus den Teilen STATUS, DESCRIPTION und REFERENCE besteht. Mittels der Objektdefinitionen werden die managebaren Objekte eindeutig festgelegt. Das OBJECT TYPE MACRO dient zur exakten Beschreibung von managebaren Objekten.

Tabellen–Definitionen: Die Tabellen–Definition beschreibt zweidimensionale Tabellen zur Darstellung komplexer Informationen.

Notifikations-Definitionen: Eine Notifikation kann ein Trap[6] oder ein *Confirmed Event*[7] sein. Der Makro (NOTIFICATION TYPE MACRO) beschreibt den Aufbau und den Inhalt.

neue Datentypen: Mit SNMPv2 sind eine Reihe neuer Datentypen eingeführt worden, wie z. B. ein 64–bit Zähler (Counter64) und ein 32–bit unsigned integer (Uinteger32).

2.4.4 Kritik an SNMPv2

Durch neue ASN.1 Makros und Protokollfunktionen ist SNMPv2 in den wesentlichen Teilen einfach und klar strukturiert (gut für die Implementierung und Portierung) geblieben. Jedoch ist durch das Hinzukommen zahlreicher neuer Konfigurationsparameter (MIB View, Security Parameter, Party Parameter) die Komplexität des gesamten "Verwaltungsapparats" erheblich gestiegen.

Außerdem ist durch zusätzliche Verschlüsselungs– und Authentifizierungsmechanismen eine hohe Grundlast entstanden, so daß die Bandbreite für die eigentlichen Nutzdaten reduziert wurde.

[6] Agent sendet dem Manager eine Message über ein unvorhergesehenes Ereignis

[7] Manager übermittelt in einer Manager–to–Manager Kommunikationsbeziehung dem anderen Manager ein unvorhergesehenes Ereignis

In bezug auf die Migrationszeit wirken sich die grundsätzlichen Unterschiede zwischen den "alten" SNMPv1 und den "neuen" SNMPv2 Standard MIBs nachteilig aus.

In der Praxis herrscht unter den Herstellern und Anwendern über die neuen Protokollspezifikationen große Verwirrung. Vor allem im Punkt Portierung stellt sich schnell die Frage: "Wie steige ich von Version 1 auf Version 2 um ?". Diese Situation trägt schließlich dazu bei, daß SNMPv2 sich bis jetzt noch nicht als Standard etabliert hat.

2.5 Anhang: Abstract Syntax Notation One (ASN.1)

2.5.1 Was ist ASN.1 ?

ASN.1 *(Abstract Syntax Notation One)* ist ein Beschreibungsmittel (Sprache) zum Aufbau von komplexen Datenstrukturen und Informationsgebilden. Sie dient zur *allgemeingültigen, herstellerübergreifenden, hardwareunabhängigen* Beschreibung von Daten. Die zu administrierenden Daten werden als Objekte (Object Type) bezeichnet. Die Datenobjekte werden durch Namen und Attribute charakterisiert. Die Namen werden zur eindeutigen Identifizierung der zu managenden Daten benutzt.

2.5.2 Notwendigkeit einer formalen Sprache

Sei die Ausgangssituation ein heterogenes Netz mit einer Vielzahl verschiedener Datendarstellungen in der Applikationsschicht. Dann braucht jeder Netzrechner Konversationsprogramme, um beliebige Kommunikation zu ermöglichen. Dies führt zu einem erheblichen Aufwand an Konversationsprogrammen (quadratischer Aufwand), der sich durch Einsatz einer standardisierten Datendarstellung entscheidend reduzieren läßt.

2.5.3 ASN.1 Objekte und Datentypen

ASN.1 unterscheidet drei Qualitäten von Objekten:

1. **Types:** beschreiben Datenstrukturen
2. **Values:** sind die Instanzen, Verwirklichungen oder Variablen eines Typs
3. **MACROS:** können die aktuelle Grammatik beschreiben, bzw. verändern; definieren die Grammatik für die MIB Objekte und werden in die MIB importiert

In ASN.1 gibt es zwei SNMP–spezifische Datentypen:

1. **UNIVERSAL TYPES:**
   Es wird zwischen *einfache Typen*, wie
   - INTEGER
   - NULL (Platzhalter)

- OCTET STRING (besteht aus null oder mehreren Octets, jedes Octet besteht aus einem Wert zwischen 0..255)
- OBJECT IDENTIFIER (definiert die Position des Objekts im MIB–Baum)

und *zusammengesetzten Typen*,

- SEQUENCE
- SEQUENCE OF

die zur Konstruktion von zweidimensionalen Tabellen verwendet werden, unterschieden.

2. **APPLICATION–WIDE TYPES:**

- COUNTER: stellt eine positive ganze Zahl dar; zyklischer Zähler; wenn das Maximum erreicht ist beginnt er wieder von Null
- GAUGE: stellt eine positive ganze Zahl dar; wenn das Maximum erreicht ist, bleibt er stehen bis ein Reset erfolgt
- TIMETICKS: zählt die Zeit in 100–stel Sekunden
- NETWORK ADDRESS
- IP ADDRESS

2.5.4 Einfache Beispiele

1. **Knoten im Managementbaum**
   Um einen Knoten (z.B. internet) im Managementbaum zu definieren, wird geschrieben:

```
internet OBJECT IDENTIFIER ::=
                  { iso org(3) dod(6) 1}
```

   Diese Spezifikation beschreibt den Pfad durch den Managementbaum bis hin zu der Stelle wo der neue Knoten (hier `internet`) eingefügt werden soll: Von der Wurzel zum `iso`–Knoten, von dort zum dritten Nachfolger `org(3)`, von dort zum sechsten Nachfolger des `org`–Knotens, dem `dod`–Knoten und schließlich soll `internet` als erster Nachfolger des `dod`–Knotens definiert werden. Darauf aufbauend können weitere Knoten definiert werden wie z.B.

```
mgmt  OBJECT IDENTIFIER ::= { internet 2 }
mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }
ip    OBJECT IDENTIFIER ::= { mib-2 4 }
```

   oder es können Objekte wie z.B.

```
ipInReceives   OBJECT IDENTIFIER ::= { ip 3 }
ipRoutingTable OBJECT IDENTIFIER ::= { ip 21 }
```

   definiert werden.

2. **Struktur definieren**
   Um zu zeigen, daß eine Struktur mit dem Namen `ContentLength` vom Type Integer ist, wird in ASN.1 folgendermaßen beschrieben:

```
ContentLength ::= INTEGER
```

   Um weiter zu sagen, daß `length` eine Variable (Instanz) der Struktur `ContentLength` ist, und um ihr einen Wert zuzuordnen wird in ASN.1

```
length ContentLength ::= 100
```

   gesetzt.

   Im allgemeinen werden solche ASN.1 Beschreibungen *modules* genannt und haben folgendes Aussehen:

```
≪ module ≫ DEFINITIONS ::= BEGIN
≪ linkage ≫
≪ declarations ≫
END
```

   Der ≪ module ≫ Term benennt das Modul. Mehrere Module können in einer Library zusammengefaßt werden und mit ≪ linkage ≫ angesprochen werden.

2.6 Literaturverzeichnis

[Ros88] [Fra91] [HG94] [Abe93] [JS93]

Kapitel 3

SNMPv2 Managementprotokoll

Raed Abdallah

3.1 Vorbemerkungen

3.1.1 Inhalt

Dieses Kapitel ist eine Ergänzung der vorherigen Kapitel über SNMP und beschäftigt sich vor allem mit der Funktionsweise eines Managementprotokolls. Außerdem werden folgende Fragen detalliert besprochen:

- welche Befehle benutzen die Managementdiensten, um Netzobjekten zu verwalten ?
- mit welchen Transportabbildungen ist das Protokoll vertraut ?
- welche neuen Befehle unterscheiden das Protokoll der Version 2 von dem der Version 1 ?

3.1.2 Was ist ein Protokoll ?

Ein **Protokoll** ist ein Satz von Regeln, die zum Austausch von Informationen zwischen Kommunikationspartnern eines Netzwerkes benutzt werden. Mehrere solche Protokolle bilden zusammen eine **Protokollfamilie**. Ein Beispiel dafür ist die TCP/IP Internet-Familie.[1]

3.1.3 Wozu Managementprotokolle ?

Bei den verschiedenen Managementdiensten werden Objekte aus der MIB des Agenten erfragt bzw. vertauscht. MIB-Objekte sind bekannterweise im ASN.1 geschrieben und beschreiben Datenstrukturen. Daher braucht man Regeln, sog. BER's (Basic Encoding Rules), die diese Objekte in Bits und Byts kodieren, d.h. sie setzen Managementbefehle in ausführbare Objekte um.

⇒ Basic Encoding Rules sind Regeln, die die durch ASN.1 beschriebenen Objekte in OCTET's (Bytes) kodieren.

3.1.4 SNMP in TCP/IP

Ursprünglich wurde SNMP für die TCP/IP-Familie entwickelt, es ist aber von seinem Konzept her nicht an bestimmte Transportprotokolle gebunden. SNMP setzt direkt auf der Transportebene der Internetprotokolle auf und verwendet das UDP (User Datagram Protocol). UDP vergibt jeden Rechner im Netz eine Portnummer, über die der gesamte Datenaustausch erfolgt. Für häufig benutzte Anwendungsbefehle stehen feste Portnummern, die in RFC (Request For Comment) veröffentlicht sind.

Richtung	Funktion	UDP-Port
NM-Station[2]⇒ Agent	SNMP-Nachricht	161
Agent ⇒ NM-Station	SNMP-Trap	162

3.2 SNMP Kommandos und Eigenschaften des Protokolls

3.2.1 SNMP Kommandos

Mit seinen 5 Basisfunktionen hat SNMP nur wenige Anforderungen an die Netzobjekte, was es den Herstellern erleichtert, Netzwerk Management zu unterstützen. Hier ist darauf hinzuweisen, daß sich solch ein einfaches Protokoll eher als Standard durchsetzen wird.

get_Request Ermöglicht die Abfrage einer bestimmten Variable in der MIB eines Agenten

get_Next_Request Meistens "the powerful command" genannt, denn es ermöglicht die Abfrage des Wertes der lexikographisch nächsten Variable, get_Next wird für Tabellen- und Baumsuche verwendet

set_Request Dieses Kommando übergibt den im get_Request angeforderten Wert

Trap Meldung eines Agenten an die NM-Station wegen eines außergewöhnlichen Ereignisses

Bemerkung

Die Request-kommandos (`get`, `getNext` und `set`) werden von der NM-Station an einem Agenten erstellt, der seinereits mit einem `get_Response` antwortet. Ein Trap wird im Gegensatz dazu von dem Agent erstellt, wird aber von der NM-Station nicht bestätigt.

[1] Die Internet-Familie wird meistens TCP/IP-Familie genannt, da TCP und IP die wichtigsten Protokolle der Familie sind

[2] NM-Station = NetzwerkmanagementStation

3.2.2 Eigenschaften des Protokolls

asynchrones Frage-/Antwort-Protokoll Eine Netzwerkmanagement-Einheit muß nicht auf die Antwort der abgefragten Einheit warten, sie kann also weitere Abfragen erstellen oder inzwischen was anderes tun.

symmetrisches Protokoll für jede Abfrage gibt es eine Antwort.

Abfragestrategien: Trap und Polling es stellt sich immer die Frage, ob außergewöhnliche Ereignisse durch einen Interrupt an die NM-Station gemeldet werden (`trap`) oder ob diese periodisch nach außergewöhnlichen Ereignissen (polling) fragt. SNMP unterstützt unter anderem eine gemischte Verfahrensweise, die sogenannte Trap-directed Polling, dabei wird nach jeder Trap-Meldung eine Polling erstellt, um festzustellen, warum dieses außergewöhnliche Ereigniss entstanden ist.

3.3 Die SNMP community

3.3.1 community-Aspekte

Mehrere Agents und NM-Stationen schließen sich zu einer Kommunikationsgruppe zusammen, mehrere solche Gruppen bilden eine "community". Kommunikation zwischen Netzwerkmanagementeinheiten kann nur mit Zugehörigkeitsnachweis der selben community folgen. Daher hat eine SNMP community folgende Aspekte:

1. **communityName**: Jede community hat einen eindeutigen Namen, der durch einen Oktett-String dargestellt ist (siehe 2.1.4).

2. **Authentifizierungsdienste**: Sie ermöglichen es einem Agenten, die Zugriffsrechte auf seine MIB zu begrenzen.

3. **Zugriffstaktik**: Der Agent kann die verschiedenen NM-Stationen verschiedene Zugriffsrechte vergeben. Er benutzt dafür zwei Aspekte:

   (a) *SNMP MIB view*: Damit wird nur ein Teilbaum des MIB-Baumes dem Abfragenden zur Verfügung gestellt.

   (b) *SNMP access mode*: Ist ein Element $\in$ { `READ-ONLY`, `READ-WRITE` }, ein solcher `access mode` ist schon für jede community vordefiniert.

4. **Proxy-Dienst** Wie aus dem 2.1.42. Kapitel schon bekannt ist, unterstützt SNMP das Prinzip von Proxies, d.h. ein Agent kann als Stellvertreter anderer NM-Einheiten im Netz wirken.

3.4 Zugriffsmechanismen

3.4.1 serial-access Technik

Diese Zugriffstechnik nutzt die Eigenschaft der MIB aus, daß sie lexikographisch geordnet ist, d.h. bis auf die Wurzel und die Blätter des MIB-Baumes hat jedes Baumobjekt einen eindeutigen Vor- und Nachfolger. Um eine Variable anzusprechen, muß der Objekt-Identifier mit einem `.Index` konkateniert werden. Um die Variable mit dem Objekt-Identifier

`sysDescr` *equals* `1.3.6.1.2.1.1.1`

anzusprechen, muß also folgendes eingegeben werden:

`get_Next_Request(sysDescr)`

Der Agent stellt zunächst eine `get_Response` mit dem Namen und Wert der lexikographisch kleinsten Variable unter dem Teilbaum `sysDescr`. Das ist gerade die Variable `sysDescr.0` mit dem dazugehörigen Wert. Jetzt kann die Abfrage mit dem neuen Parameter von vorne anfangen, also

`get_Next_Request(sysDescr.0)`

usw.

Um Tabellen mit k Spalten anzusprechen wird die Funktion `get_Next_Request` mit der entsprechenden Zahl von Parametern aufgerufen.

Aufruf:

`get_Next_Request(`$\texttt{Name}_1$`,` $\texttt{Name}_2$`, ...,` $\texttt{Name}_k$`)`

Antwort:

`get_Response(`$\texttt{Name}_1$`.ObjID=`$\texttt{Wert}_1$`,`
$\texttt{Name}_2$`.ObjID=`$\texttt{Wert}_2$`, ...,` $\texttt{Name}_k$`.ObjID=`$\texttt{Wert}_k$`)`

3.4.2 random-access Technik

Bei dieser Technik handelt es sich um folgende Art von Tabellen:

- Es existiert eine Tabelle aus $\geq$ NULL Reihen.
- Die Spalten der Tabelle unterscheiden sich durch einen Index, sog. SpaltenIndex.
- Jedes Spaltenobjekt hat einen eindeutigen Objekt-Identifier, der bis auf einen SpaltenIndex überall in der gleichen Reihe identisch ist.

Die Konvention bei SNMP ist es, Objekt-Identifier mit dem Index zu konkatenieren, um einen Instance-Identifier zu erzeugen.

Wir betrachten jetzt ein kompliziertes Beispiel einer TCP-Verbindungstabelle, die aus fünf Spalten besteht (Tabelle 3.1).

Jeder Instance-Identifier eines Objektes der Tabelle wird von SNMP in der folgenden Form erzeugt:

`x.i.(tcpConnLocalAddress).(tcpConnLocalPort).(tcpConnRemAddress).(tcpConnR`

wobei

- `x` = Objekt-Identifier von `tcpConnEntry`,
- `i` = SpaltenIndex
- `Name` = Wert vom *Name*

Die Objekte der Tabelle 3.1, haben dann Instance-Identifier, wie sie in Tabelle 3.2 gezeigt werden.

Der Zustand der TCP-Verbindung für die 1.Reihe ist dann,

`tcpConnState`.10.0.0.99.12.9.1.2.3.15

insgesamt also,

1.3.6.1.2.1.6.13.1.1.10.0.0.99.12.9.1.2.3.15

3.5 SNMP-PDU's

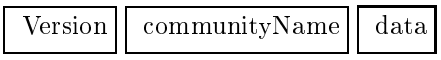
3.5.1 SNMP-Message

Der Informationsaustausch zwischen Netzwerk Management Einheiten wird in Form von Messages erfolgen. Eine SNMP-Message ist eine Folge oder ein dreier Tupel; (version, communityName, data), wobei

version ist vom Typ INTEGER, für MIB II, RFC 1185, steht hier der Wert 0.

communityName ist vom Typ OCTET STRING, und enthält typischerweise einen ASCII Namen.

data ist ein Datenfeld vom Typ ANY und enthält eines der fünf SNMP-PDU's.



⇒ ASN.1-Schreibweise für SNMP-Message:

```
SNMP-Message ::= SEQUENCE {
      version      INTEGER,
      community OCTET STRING,
      data         ANY }
```

3.5.2 Das Datenfeld (PDU-Anteil)

Erst hier wird der wirkliche Befehl bekanntgegeben. PDU (`Protocol Data Unit`) kennt fünf Befehle, für die es unterschiedliche Werte in der PDU-Folge gibt. Die PDU-Folge ist ein fünf-Tupel: (PDU-Typ, request-id, Error-status, Error-index, Variable-bindings).

⇒ ASN.1-Schreibweise für `get_Request-PDU`:

```
get_Request-PDU ::= IMPLICIT SEQUENSE {
    request-id          RequestID,
    error-status        ErrorStatus,
    error-index         ErrorIndex,
    variable-bindings VarBindList }
```

wobei `variable-bindings` eine Liste der abgefragten Variablen darstellt.

⇒ `Variable` ist ein Paar (Name, Wert); Name ist der Objekt-Identifier und der Wert wird je nach Anfrage entsprechend belegt.

Fehlerart	Error-index	PDU-Befehl	PDU-Typ
noError	0	get_Request	0
tooBig	1	get_Next_Request	1
noSuchName	2	get_Response	2
badValue	3	set_Request	3
readOnly	4	get_Bulk_Request	4
genError	5	Inform_Request	5
		trap	6

Tabelle 3.3: Fehler- und die PDU-Typ-/Einträge

3.5.3 Transport von Massages

Sendefunktionen

- Der Sender erzeugt die notwendigen PDU-Kommandos als ASN.1-Objekt.
- Dieses ASN.1-Objekt wird mit dem communityName der Source- und Destination-Adresse an einem Authentifizierungsdienst übertragen.
- Der Authentifizierungsdienst erzeugt ein weiteres ASN.1-Objekt.
- Diese Objekt-Struktur wird dann nach BER kodiert und seriell an das Transportprotokoll übertragen.

Empfangsfunktionen

- Das ankommende Datagramm wird auf Gültigkeit der übertragenen Daten überprüft, ist es ungültig, dann wird das Datagramm vernichtet und nichts weiteres getan.
- Im Falle einer gültigen Übertragung wird die Versionsnummer der SNMP-Message mit der eigenen SNMP-Versionsnummer verglichen. Wenn sie nicht identisch sind, wird das Datagramm auch vernichtet und nichts weiteres getan.
- Bei Übereinstimmung wird der `communityName` und die im ASN.1-Message-Objekt enthaltenen Daten zusammen mit der Source- und Destination-Adresse an dem Authentifizierungsdienst übertragen.
- Der Authentifizierungsdienst gibt entweder ein ASN.1-Objekt oder ein Authentifizierungsfehler zurück, falls letzteres eintritt wird dieser Fehler registriert, ein Trap generiert und das Datagramm wird vernichtet.

- Das ASN.1-Objekt, das vom Authentifizierungsdienst zurückgegeben wird, wird mit der Orginal-PDU verglichen; bei Übereinstimmung wird die PDU je nach `communityName` und SNMP access policy, also ein Element $\in$ {READ-ONLY, READ-WRITE}, verarbeitet.

- Die Message, die aufgrund dieser Funktion erzeugt wird, wird automatisch an die Destinationadresse der eingegangenen Message zurückgeschickt.

3.5.4 Transportabbildungen

Connectionless Transport Service (CLTS)

Zur Übertragung von SNMP-Messages werden vorallem verbindungslose Mechanismen benutzt, denn SNMP in der Internet-Protokollfamilie das UDP benutzt, das verbindungslos funktioniert. Bei dieser Art von Übertragung wird das Datagramm in Datenpackete geteilt, die dann seriell an die Zieladresse geschickt werden.

Connection-Oriented Transport Service (COTS)

SNMP kann trotzdem verbindungsorientiert betrieben werden[3], was sich aber als weniger vorteilhaft ergeben hat. CMIP (Common Management Information Protocol), ein Protokoll, das ursprünglich SNMP ersetzen sollte, arbeitet verbindungsorientiert, wegen seiner aufwendigen Struktur blieb aber von den Netzwerktreibern unbeliebt.

3.6 SNMPv2 und Blick in die Zukunft

3.6.1 SNMPv2-PDU's

SNMPv2 hat unter anderem in dem Protokoll selbst Unterschiede zu dem SNMPv1, denn SNMPv2 soll das Netz so gut wie möglich entlasten, außerdem soll SNMPv2 nicht nur als Client/Server-Modell funktionieren, sondern auch noch Manager-to-Manager Kommunikation ermöglichen. Um diese neue Aufgaben zu beherrschen, wurde das Protokoll um die zwei folgende Kommandos ergänzt:

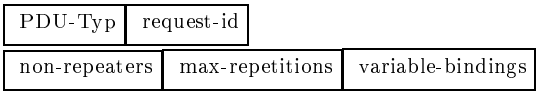
get_Bulk_Request-PDU

Dieses Kommando stellt eine Verallgemeinerung der `get_Next_Request-PDU` dar, es soll die Belastung des Netzes reduzieren, indem zunächst nicht für jede `get_Next_Request-PDU` eine `get_Response-PDU` generiert wird und damit das Netz total belastet, sondern bei jedem `get_Bulk_Request` werden zwei Parameter geschickt:

- `non-repeaters`: gibt an, wieviele Variablen aus der Variable-bindings nicht wiederholt werden müßen.

- `max-repetitions`: funktioniert wie ein Zähler für die Häufigkeit der Wiederholung von `get_Next_Request-PDU's`.

Am Ende der Abarbeitung wird ein einziger `get_Response`-Befehl für alle abgefragten Variablen erstellt.



Inform_Request

Dieses Kommando ist für die Manager-to-Manager Kommunikation vorgesehen. Da bei SNMPv2 ein NM-Station als Manager/Agent-Einheit funktionieren kann und eine NM-Station nicht wissen kann, ob die Netzwerk-Einheit, mit der sie gerade kommuniziert, eine Agent oder auch eine NM-Station ist, wird solches PDU zu der Zieladresse `SNMPv2EventNotifyTable`, die in der Manager-to-Manager MIB definiert ist, geschickt. Ein Inform_Request-PDU hat die gleiche Format wie get_Request-PDU.

3.6.2 Blick in die Zukunft

Aus dem bisher gesagten wird deutlich, daß SNMP einfache Kommandos verwendet, was natürlich auch bedeutet, daß es weniger fehleranfällig ist. Mit get kann SNMP Variablen der MIB abfragen, mit get-Next kann sie sogar einen ganzen MIB-Baum durchsuchen. SNMP kann diese Variablen mit dem Set-Befehl leicht ändern. Für außergewöhnliche Ereignisse gibt es außer Trap-Meldung das Polling und das Trap-directed Polling. In SNMPv2 würde vieles im Bezug auf Sicherheit erreicht $\Rightarrow$ Partykonzept. Darüberhinaus hat man bei SNMPv2 die Belastbarkeit des Netzes reduziert, was ein wesentlicher Nachteil von SNMP darstellt.

Vom OSI wird um 1995/96 ein OSI-Management erwartet, das das Netz weiter entlasten soll, indem es mehr Last auf die lokalen CPU's der Netzobjekte schiebt.

Zur Zeit ist SNMP *das* Management Protokoll, das nicht nur im Internet, sondern auch in vielen Netzen realisiert ist.

3.7 Literaturverzeichnis

[LF89] [Sta93c]

[3] dazu muß SNMP TCP statt UDP benutzen

tcpConnState	tcpConnLocalAddress	tcpConnLocalPort	tcpConnRemAddress	tcpConnRemPort
(1.3.6.1.2.1.6.13.1.**1**)	(1.3.6.1.2.1.6.13.1.**2**)	(1.3.6.1.2.1.6.13.1.**3**)	(1.3.6.1.2.1.6.13.1.**4**)	(1.3.6.1.2.1.6.13.1.**5**)
5	10.0.0.99	12	9.1.2.3	15
2	0.0.0.0	99	0	0
3	10.0.0.99	14	89.1.1.42	84

Tabelle 3.1: TCP-Verbindungstabelle

tcpConnState	tcpConnLocalAddress	tcpConnLocalPort	tcpConnRemAddress	tcpConnRemPort
(1.3.6.1.2.1.6.13.1.1)	(1.3.6.1.2.1.6.13.1.2)	(1.3.6.1.2.1.6.13.1.3)	(1.3.6.1.2.1.6.13.1.4)	(1.3.6.1.2.1.6.13.1.5)
x.1.10.0.0.99...	x.2.10.0.0.99...	x.3.10.0.0.99...	x.4.10.0.0.99...	x.5.10.0.0.99...
x.1.0.0.0.0.99.0.0	x.2.0.0.0.0.99.0.0	x.3.0.0.0.0.99.0.0	x.4.0.0.0.0.99.0.0	x.5.0.0.0.0.99.0.0
x.1.3.10.0.0...	x.2.3.10.0.0...	x.3.3.10.0.0...	x.4.3.10.0.0...	x.5.3.10.0.0...

Tabelle 3.2: Instance-Identifier der Objekte der `tcpConnTable` Tabelle

Abbildung 4.1: OSI Architekturmodell

in anderen Knoten. Insbesondere mit dem System, das als Netzkontrollzentrum funktioniert.

- Layer Management Entity (LME):
  Befaßt sich mit der Verwaltung von Objekten, die zu einer bestimmten Schicht des OSI-Schichtenmodells zugeordnet werden können.

- Management Information Base (MIB):
  Ist definiert als Ablage der Managementinformationen eines Knoten, der zu einem Netzwerk gehört.

Die Abbildung 4.1 zeigt uns die Beziehung unter diesen Komponenten. Man kann die SMA-Entity logisch definieren, wie eine zusammenhängende Menge von Applications Services Elements (ASEs). Der SMASE stellt mehrere Services bereit, auf die sowohl der Netzwerkmanager als auch die Anwendungen, die Netzwerkmanagementfunktionen implementieren, zurückgreifen können.

Zwei ASEs sind zum allgemeinen Einsatz in verschiedenen Anwendungen entwickelt worden. ACSE dient zur Herstellung von Verbindungen und ROSE zur Übertragung der PDUs.

Der SMAP kann in zwei Rollen auftreten: als Managementsprozeß oder als Agentprozeß. Dabei verwaltet der Agentprozeß einerseits unmittelbar die ihm zugeordneten verwalteten Objekte, auf der anderen Seite kommuniziert er mit einem oder mehreren Managementprozeßen, die die eigentlichen Managementfunktionen wahrnehmen. D.h., der Manager fordert Informationen von den managed Systemen auf, und schickt ihnen Kommandos (Operationen) zur Ausführung. Die Manager-Rolle wird von einem System gespielt, das als Netzwerkkontrollzentrum funktioniert.

Die Kommunikation zwischen Manager und Agent wird über die Funktionen des Common Management Information Service (CMIS) und des Common Management Information Protocol (CMIP) abgewickelt.

Basis des Managementmodells ist die Benutzung von objektorientierten Prinzipien. Ein Objekt ist durch seine Attribute, Operationen, die auf ihn ausgeführt werden können, Nachrichten, die es senden kann, und seine Beziehungen zu anderen Objekten definiert. Es kann irgendeine Hardware, Software, oder logische Netzwerkkomponente sein. Objekte vertreten Resourcen, auf die man Operationen ausführen kann.

Eine Managementoperation wird auf ein Objekt dadurch angewandt, daß an das Objekt eine Nachricht geschickt wird, die sowohl die Art der Operation als auch Parameter dazu enthält. Das Objekt interpretiert die Nachricht und führt sie aus, bzw. weist die Operation zurück. Nur diejenigen Operationen, die für das jeweilige Objekt definiert sind, sind dem Netzwerkmanager vorhanden.

4.2.2 OSI Funktionale Bereiche

Die OSI Managmentdokumente teilen die Aufgaben des Netzwerkmanagements in fünf funktionale Bereiche (SMFA) auf. Die SMFA beschreiben die allgemeinen Anforderungen, die zur Bewältigung der Aufgaben der einzelnen Funktionalebereiche erfüllt werden müssen. Die Bereiche sind: Fehlerbehandlung, Kostenberechnung, Konfiguration und Namensbereichsbestimmung, Leistungsüberwachung, Sicherheitsverwaltung.

4.2.3 Systems Management Functions (SMF)

Die SMFAs beschreiben eine erweiterte Netzwerkmanagement Verantwortung. Jeder von diesen Bereichen hat die Benutzung von bestimmte Funktionen zur Folge. Es gibt eine beträchtliche Überlappung in diesen Funktionen. Es gibt je nach SMFA keine Zuordnung der Funktionen. Jeder SMF-Standard definiert die Funktionalität zur Unterstützung der SMFAs-Aufforderungen.

Eine gegebene SMF kann die Aufforderungen für einen oder mehrere Funktionale Bereiche unterstützen. Von einem anderen Blickpunkt, jeder funktionale Bereich fordert mehrere SMFs auf.

Sie werden in einer anderen Ausarbeitung genau behandelt.

4.3 Informationsmodell

Die Grundlage jedes Netzwerkmanagementsystems ist eine Datenbank, die die Informationen über die verwalteten Ressourcen und Elemente, enthält. Im OSI System Manager Informationsmodell werden die managed Ressourcen bei managed Objekten dargestellt. Die MIB ist eine strukturierte Menge dieser Objekte.

Switches, WS, PBX, Portcards, Multiplexer sind Beispiele von Ressourcen. Andere Beispiele (Software) sind: Queuingprogramme, Routingsalgoritmen.

4.3.1 Grundlegende Begriffe des OSI Management Information Modell

Die Spezifikation der OSI MIB und Structure of Management Information benutzt OOD Prinzipien, die die modularische Ableitung oder Zufügung und Wiederverwendung neuer managed Objektenklassen und Funktionen ermöglichen. Die Spezifikation schreibt nicht vor, wie die MIB implementiert wird.

Managed Objekt

Ein managed Objekt ist durch seine Attribute definiert, die Operationen, die auf ihm ausgeführt werden können, Anmeldungen, die es ausgeben kann, und seine Beziehungen mit anderen verwalteten Objekten. Zur Gliederung der MIB-Definition ist jedes managed Objekt eine Instanz einer managed Objektklasse.

Attributte

Die aktuellen Datenelemente des managed Objektes nennt man Attribute. Jedes Attribut stellt eine Eigenschaft des Objektes dar: z.B. operationale Charakteristik, aktuellen Zustand, operationale Bedingungen.

Abbildung 4.2: Containment und Benennung

Ein untergeordnetes managed Objekt darf nur in einem Oberobjekt enthalten sein. Dadurch wird die MIB-Struktur als ein Baum aufgebaut.

Bei der Objektklasse wird ihr Bezeichner im Registrations-Baum als eindeutiger Objektbezeichner eingetragen. Jeder Bezeichner ist eine Ganzzahlsequenz, die durch den Registration-Baum navigiert.

Das Benennungschema für Objektinstanzen hat drei Vorschriften:

- Namensattribut:
  Jede Managed Objektklasse schließt ein Attribut zur Instanzbenennung dieser Klasse ein.

- RDN:
  Der Relativ Distinguished Name einer Objektinstanz

Abbildung 4.3: Management Information

4.3.3 Systemmanagementoperationen

Der Standard definiert die auf das Objekt ausführbaren Operationen. Sie werden bei einer Entität ausgeführt. Man schickt eine Nachricht an das Objekt, wobei ein Protokoll benutzt wird. Es gibt zwei Operationskategorien:

1. Objektorientierte Operationen
   Die folgenden Operationen können auf das ganze Objekt angewendet werden.
   - Create
   - Delete
   - Action
2. Attributorientierte Operationen
   - Get attribute value
   - Replace attribute value
   - Set attribute value to default
   - Add member to a set-valued Attribute
   - Remove member from a set-valued attribute

Als Ergebnis der Operation, schickt das Objekt dem Netzwerkmanager oder dem Logfile eine Nachricht, die den Attributbezeichner und seine angeschlossenen Werte enthält, für diejenige die eine erfolgreiche Operation aufweisen, und eine Fehlermeldung für diejenigen, bei denen die Operation nicht erfolgreich war.

4.3.4 Management Information Definition

Der Standard definiert einige Grundklassen, Attribute und Nachrichten, die bei dem MIB-Aufbau benutzt werden können. Die sind: Generische Attribute, Spezifische Attribute, Nachrichtentypen, Managed Objektklassen.

Generische Attribute

Diese Attribute werden definiert, weil Zahlenoperationen in vielen Kontexten sehr nutzlich sind. Generische Attribute kann man sich als Attributetypen oder Makros vorstellen, die zur Konstruktion von spezifischen Attributen benutzt werden. Sie teilen sich in 2 Kategorien auf:

- Zähler (Counters)
- Gauge

Zähler oder Counter

Zähler ist eine Abstraktion eines grundlegenden Zahlungsprozeßes. Der Zähler ist immer eine positive ganze Zahl, die nur zunehmen kann. Man muß aber einen maximalen Wert definieren, bei dem der Zähler wieder von Null starten kann. Man definiert zwei Zählertypen:

- Non Setable: Nicht setzbarer
- Setable : Setzbarer

Ein nicht-setzbarer Zähler kann durch eine Managementoperation nicht verändert werden. Er ändert sich als eine Antwort auf ein Ereignis. Dieser Zähler eignet sich für Situationen, bei denen mehrere Management Stationen auf den Zähler Zugriff haben. Da der Zähler nicht gesetzt werden kann, können die Management Stationen bei der Benutzung des Zählers nicht einander stören.

Ein setzbarer Zähler kann bei einer Management Operation verändert werden. Deshalb eignet er sich nur zur

Abbildung 4.4: Templates

4.4 CMIS UND CMIP

Die Hauptfunktion des OSI Management Systems ist der Austausch von Managementinformationen zwischen zwei Entitäten - Manager und Agent- mittels eines Protokolls. Die Services, die zur Implementierung dieser Aufgabe vorhanden sind, werden CMIS genannt. Im CMIS werden Funktionen zur Übermittlung von Management Informationen beschrieben. Der Transport der Informationseinheiten (PDUs) erfolgt über das CMIP.

Die Abbildung 4.5 zeigt den Kontext von CMIS und CMIP. Beide zusammen gestalten das CMISE. Das CMIS stellt sieben Services zur Ausführung von Management Operationen bereit. Außerdem benötigen die CMISE Benutzer einen Dienst zur Verbindungsherstellung. Dieser Service ist durch das ACSE bereitgestellt. Das ist ein durch sicherer Service, kein CMIP ist dabei eingeschlossen. Für die Management Operation Services benutzt die CMISE ein CMIP zum Austausch von PDUs. Seinerseits veläßt sich das CMIP auf die ROSE

Abbildung 4.5: Schichtung von CMISE

Services. Beide, ROSE und ACSE verlassen sie sich auf die Darstellungservices.

CMIS ist im Sinne des OSI Basic Reference Model eine Anwendung der Schicht sieben und kann daher zur Abwicklung der Kommunikation seinerseits OSI Basisdienste, nämlich die Remote Operations Service Elements (ROSE) benutzen. ROSE definieren ein relativ einfaches Anfrage/Antwort Verfahren.

Jeder Dienst wird in einer Konversation zwischen Manager und Agent in fünf Varianten benutzt, nämlich als `Request`, `Indikation`, `Response`, `Confirmation`, und `Error`, die Primitive genannt werden.

Die strukturellen Einrichtungen von CMIS sind:

- Linkage
- Auswahl (Managed Object Selection)

Verbindung (Linkage)

Bei der Linkage Operation können mittels eines `link-id` Parameters vielfache Antworten auf Confirmed (bestätigte) Operationen angeschlossen werden. Z.B. kann M-GET eine Operation auf mehrere Objekte spezifizieren. Dann wird eine Antwort für jedes Objekt generiert und zurückgeschickt. Man braucht eine Linkage-Technik, um alle Antworten für die ursprüngliche Aufforderung wieder zusammenzusetzen.

Management Objekt Auswahl (Managed Object Selection)

Zur Auswahl eines oder mehrerer Objekten, auf die eine Operation angewendet wird, werden drei Werkzeuge benutzt:

- Scoping
  Man identifiziert das Objekt, auf das ein Filter angewandt wird. Scoping ist mit dem Verweis auf eine spezifische Managed Object Instanz definiert, auf die als Base Managed Object verwiesen wird. Dieses Managed Object ist der Startpunkt für die Auswahl eines oder mehrerer Objekte. Man benutzt dieses Base Object als Wurzel des Containment Teil-Baums.

- Filtering
  Ein Filter ist ein boolscher Ausdruck, er besteht aus einer oder mehrerer Aussagen über die Anwesenheit oder Werte der Attributen in einem "scoped" management Object. Jede Aussage ist ein Test für gleiche Wertigkeit, Ordnung, Anwesenheit oder Mengevergleich.

- Synchronization
  Da die Scope und Filter Parameter mehr als ein Objekt ergeben können, ergibt sich die Frage, in welcher Ordnung die Operation angewandt wird. Da die resultierende Zuordnung der Objektinstanzen bei der Anwendung des Filters nicht spezifiziert ist, ist sie deswegen nicht benutzbar. Dann wird der Synchronizationservice angewendet. Es gibt zwei Typen von Synchronization:

  – Atomic
    Alle für die Operation ausgewählten Objekte werden überpruft, ob sie für eine erfolgreiche Operation fähig sind. Wenn ein oder mehrere Objekte nicht fähig sind, die Operation auszuführen, dann wird diese Operation von keinem Objekt ausgeführt.

  – Best-Effort
    Alle für die Operation ausgewählte Objekte werden zu ihrer Ausführung aufgefordet.

Scope, Filter, Synchronisation werden wie folgt angewandt:

- Wenn der Scope-Parameter anwesend ist, ist das Scope das Base Object. Ansonsten ist das Scope das im Scopeparameter spezifizierte Objekt.

- Wenn der Filter-Parameter abwesend ist, dann werden alle Managed Objekte, die beim Scope eingeschlossen wurden, ausgewählt. Ansonsten werden nur diese Objekte, die durch den Filter durchsickern, ausgewählt.

- Wenn der Synchronization-Parameter abwesend ist, wird der "Best effort" Synchronizationtyp ausgeführt. Wenn nur das Base-Object ausgewählt wird, dann wird der Synchronizationsparameter übersehen. Wenn der Synchronizationsparameter anwesend ist, und das Scope mehrere Objekte ausgewählt hat, bestimmt der Synchronizationsparameter, welche Synchronizationstyp angewendet wird.

CMIS Services Kategorien

1. Association Service
   Das ist das ACSE. Auf eine ausführliche Beschreibung wird in diesem Seminar verzichtet.

2. Management Notification Service
   Hier wird nur M-EVENT-REPORT definiert, zur spontanen Übermittlung einer Ereignismeldung. Im Unterschied zu den anderen Managementoperationen ist der EventReport von dem Agent initiiert.

3. Management operations services
   M-CREATE, M-DELETE, M-GET, M-CANCEL-GET, M-SET, M-SET, M-ACTION.

4.5 Literaturverzeichnis

[Sta93e] [Sta93a]

Kapitel 5

OSI Management Funktionalität

Petra Philips

5.1 Einführung

Im Rahmen des ISO-Referenzmodells wird nicht nur die Kommunikation von Netzen standardisiert, sondern auch deren Verwaltung. Der Standard für die Verwaltung, das OSI systems management, enthält grundlegende Architekturkonzepte zur Integration der Management-Aufgaben in Kommunikationsnetzen (mit entsprechender Terminologie) und Basisfunktionen zur Realisierung des Management-Systems.

In dieser Ausarbeitung werden die bisher definierten Basisfunktionen vorgestellt. Dabei werden im nachfolgenden Teil 5.2 nach einer kurzen Einführung zuerst die wichtigsten Konzepte und Begriffe des OSI systems management kurz erläutert. Im Kapitel 5.3 werden die Funktionen ausführlicher behandelt, jedoch werden die konkreten Definitionen der Dienstprimitive nicht vorgestellt. Es wird vielmehr versucht, einen Eindruck über die Funktionalität und generelle Prinzipien zu vermitteln.

5.2 Grundbegriffe des OSI Systems-Management

5.2.1 OSI-Management-Architektur

Die OSI-Management-Architektur für eine Komponente eines Netzes baut auf dem ISO-OSI-7-Schichten-Modell auf. Wie auf Bild 5.1 zu sehen, gibt es innerhalb jeder der 7 OSI-Schichten (physical layer, ..., application layer) eine Einheit, die LME (Layer Management Entity), die für die schichtspezifischen Management-Funktionen zuständig ist. Alle LMEs greifen auf die MIB (Management Information Base) zu, welche die Management-Information der Komponente enthält. Die SMAP (Systems-Management Application Process) ist die lokale Software, die für die Management-Funktion in der Komponente verantwortlich ist. Mit Hilfe der SMAE (Systems-Management Application Entity) innerhalb Schicht 7 (application layer) erfolgt der Informations-Austausch mit Partner-SMAEs aus anderen Knoten über die CMIP (Common Management Information Protocol). Innerhalb der SMAE befinden sich die standardisierten Management-Funktionen, die in dieser Ausarbeitung ausführlicher beschrieben werden.

5.2.2 MO (Managed Object)

Eines der wichtigsten Konzepte des OSI Management-Systems ist das MO (Managed Object). Es ist die Einheit für Information und repräsentiert auf abstrakter Weise die Ressourcen des Netzes. MOs werden objektorientiert definiert, d.h. sie haben Attribute, man kann Operationen auf sie ausführen und sie können Meldungen (notifications) über ihren Zustand senden. Zusätzlich können MOs in Beziehungen zueinander stehen. Die MIB ist nichts anderes als eine strukturierte Kollektion solcher MOs.

5.2.3 SMFA (Specific Management Functional Areas)

OSI unterscheidet fünf Bereiche des Managements: fault management, accounting management, configuration and name management, performance management und security management.

5.2.4 SMFs (Systems-Management Functions)

Um die geforderte Managementfunktionalität der 5 SMFAs zu erbringen, definiert die ISO standardisierte Hilfsfunktionen, die SMFs. Bisher wurden 13 Funktionen definiert:

- object-management function
- state-management function
- relationship-management function
- alarm-reporting function
- event-report-management function
- log-control function
- security-alarm-reporting function
- security-audit-trail function

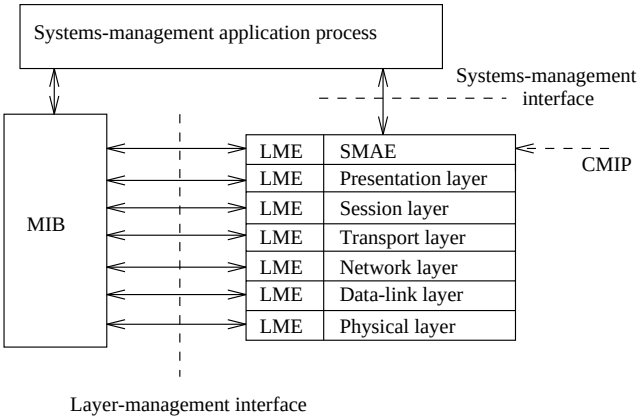


Abbildung 5.1: Modell der OSI-Management-Architektur

- access-control-management function
- accounting-meter function
- workload-monitoring function
- test-management function
- summarization function

Diese wiederum benutzen die Dienstprimitive des CMISE: `M-EVENT-REPORT`, `M-GET`, `M-SET`, `M-ACTION`, `M-CREATE`, `M-DELETE`, `M-CANCEL-GET` (siehe Bild 5.2, für Details über CMISE-Dienste siehe [Sta93a]).

Im folgenden Kapitel werden diese Funktionen einzeln ausführlicher vorgestellt.

5.3 OSI Management-Funktionen

5.3.1 Object-Management Functions

Die object-management functions bietet Dienste für die Verwaltung der MOs an. Dabei gibt es zwei Kategorien von Diensten:

pass-through services sind für Basisoperationen auf MOs zuständig, also für das Erzeugen eines neuen MOs, das Löschen eines existierenden MO und das Modifizieren der Attribute von MOs;

direct services sind für Meldungen über Änderungen der MOs zuständig.

Pass-Through Services

Da die CMISE bereits Dienstprimitive für Basisoperationen auf Objekten bereitstellt, werden die pass-through services direkt auf diese abgebildet. Daher auch ihr Name: sie machen nichts anderes als ihre Parameter an CMISE weiterzureichen (pass-through = weiterreichen). Die Korrespondenz zwischen den pass-through Diensten und den Diensten der CMISE ist in folgender Tabelle dargestellt.

Pass-Through Dienste	Dienste des CMISE
PT-CREATE	M-CREATE
PT-DELETE	M-DELETE
PT-ACTION	M-ACTION
PT-SET	M-SET
PT-GET	M-SET
PT-EVENT-REPORT	M-EVENT-REPORT

Die natürliche Frage angesichts dieser erneuten Definition (die gleichen Primitive existieren ja schon für CMISE) ist, warum definiert ISO die pass-through services überhaupt ? Der Grund dafür ist der, daß dadurch eine Entkopplung von CMIS/CMIP erreicht wird. Es besteht also durchaus die Möglichkeit, SNMP anstelle von CMIS zu nutzen, und trotzdem die OSI-Funktionen zur Verfügung zu haben.

Direct Services

Die direct services werden verwendet, um Meldungen (notifications) den Partnerinstanzen aus anderen Knoten bekanntzumachen. ISO definiert drei Meldungen, zusammen mit den verwendeten Parametern und deren Semantik:

- object-creation reporting
- object-deletion reporting
- attribute-value-change reporting

Alle drei werden mit Hilfe von `M-EVENT-REPORT` erbracht, dabei werden verschiedene Parameter weitergegebenen, welche Informationen enthalten, wie: Identifizierung des Objektes, Art der Änderung des Objektes, der Initiator der Änderung. Mögliche Initiatoren für die Änderungen sind:

- systeminterne Vorgänge,
- das lokale System (eine höhere Schicht des lokalen Systems),
- ein entferntes System.

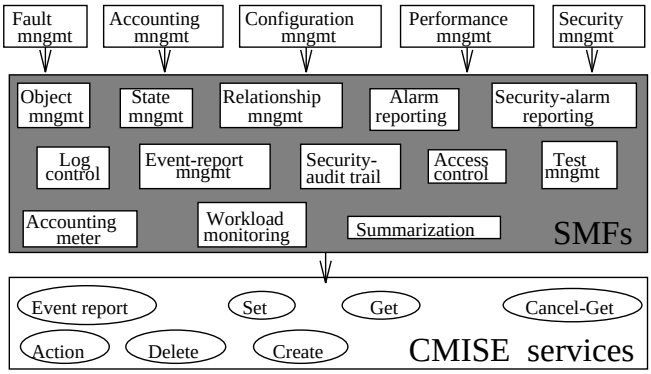


Abbildung 5.2: System-Management

5.3.2 State-Management Function

Im Bereich der state-management function wird ein Modell der Managementzustände des MO spezifiziert, und es werden Dienste für die Kontrolle der entsprechenden Attribute definiert.

Modelle der Zustände

Damit die Ressourcen überwacht werden können, werden grundlegende Zustände für die MOs definiert. Ziel ist es, Ressourcen, die nicht verfügbar sind, zu erkennen, und die Ursache dafür und mögliche Aktionen zur Beseitigung dieses Zustandes anzuzeigen.

ZUSTANDSATTRIBUTE Es gibt drei Faktoren, die die Verfügbarkeit einer Ressource beeinflussen: Betriebsbereitschaft, Nutzung und Administration. Daher definiert ISO drei Zustandsattribute – operational state, usage state, administrative state – deren mögliche Werte als Zustände in Zustandsdiagrammen beschrieben werden.

Das operational state diagram beschreibt die Betriebsbereitschaft der Ressource, d.h. ob die Ressource überhaupt installiert ist oder nicht. Es gibt 2 Zustände dafür: `ENABLED` für eine installierte Ressource, `DISABLED` für eine Ressource die nicht installiert ist.

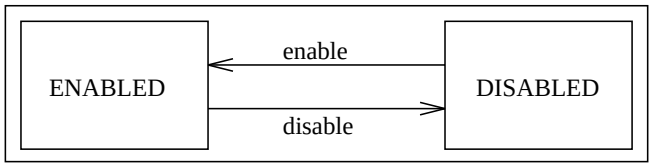


Abbildung 5.3: Operational state diagram

Das usage state diagram beschreibt die mögliche Nutzung der Ressource (ob sie überhaupt genutzt wird) und die noch verfügbare Kapazität, d.h. ob noch weitere Dienstbeantragungen ausgeführt werden können. Die möglichen Attribute sind: `IDLE`, falls die Ressource gerade nicht genutzt wird, `BUSY`, falls die Ressource gerade voll genutzt wird, so daß keine freie Kapazität für weitere Dienstanfragen frei ist, `ACTIVE`, falls die Ressource genutzt wird, aber noch freie Kapazität da ist und `UNKNOWN`, falls keine Informationen über die aktuelle Nutzung der Ressource vorhanden ist.

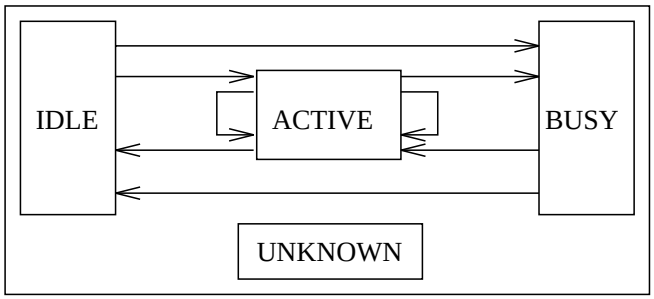


Abbildung 5.4: Usage state diagram

Das administrative state diagram beschreibt den administrativen Zustand der Ressource. Es beschreibt Ressourcen, die aus administrativen Gründen gezielt aus der Benutzung herausgenommen werden, z.B. um Reparaturen durchzuführen oder die Konkurrenz zu regeln. Hier werden drei Zustandsattributwerte unterschieden: `UNLOCKED`, wenn die Ressource freigegeben ist, `SHUTTING DOWN`, wenn sie aus administrativen Gründen gesperrt ist, jedoch die gerade anstehenden Anfragen noch bearbeitet werden und `LOCKED`, falls die Ressource keine Benutzer mehr bedienen darf.

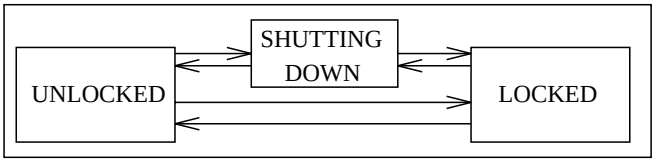


Abbildung 5.5: Administrative state diagram

STATUSATTRIBUTE Für eine genauere Beschreibung des Zustandes einer Ressource werden sogenannte Statusattribute zusammen mit ihren Werten spezifiziert:

- `ALARM STATUS` zeigt die Anwesenheit verschiedener Alarme in Verbindung mit der Ressource an.
- `PROCEDURAL STATUS` gibt Informationen darüber, in welcher Phase sich ein Objekt befindet, das eine Prozedur representiert (z.B. Ressource ist in der

Initialisierungsphase, Ressource ist in der Terminierungsphase, etc...).

- `AVAILABILITY STATUS` gibt zusätzliche Informationen über den operational state der Ressource an.
- `CONTROL STATUS` gibt zusätzliche Informationen über den administrative state der Ressource an.
- `STANDBY STATUS` gibt zusätzliche Informationen über eine Backup-Relation der Ressource (wenn diese existiert) an – siehe 5.3.3.
- `UNKNOWN STATUS` gibt an, ob der Status der Ressource bekannt ist oder nicht (boolescher Wert).

Dienste

Als Dienstprimitive zum Lesen und Ändern der Zustands- und Statusattribute werden die Dienstprimitive der object-management function (`PT-GET`, `PT-SET`) verwendet.

Für die Übertragung der aktuellen Werte der Attribute nach einer Änderung wird das Dienstprimitiv state-change-reporting-service definiert, welches mit Hilfe von `M-EVENT-REPORT` erbracht wird. Dabei wird der Parameter `event-type` als `state change` gesetzt.

5.3.3 Relationship-Management Function

Innerhalb der relationship-management function werden Modelle für Abhängigkeiten (Beziehungen) zwischen Objekten definiert, sowie Dienste, um diese Beziehungen zu verwalten.

Modelle für Beziehungen

Eine Beziehung oder Abhängigkeit zwischen Teilen eines Netztes existiert dann, wenn Operationen auf dem einen Teil den Betrieb des anderen Teiles beeinflussen. Zum Beispiel kann der Ausfall eines Druckers einen anderen Drucker aktivieren.

Dabei gibt es generelle Klassifizierungen von Beziehungen in direkte und indirekte, symmetrische und asymetrische, etc. ... In Bild 5.6 wird beispielsweise der Unterschied zwischen einer direkten und indirekten Beziehung gezeigt.

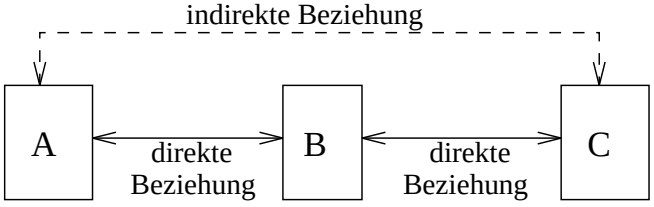


Abbildung 5.6: Direkte und indirekte Beziehungen

ISO definiert Typen von Beziehungen (relationship types), die die Natur der Beziehung zwischen MOs beschreiben. Die relationship role ist die Rolle der einzelnen Objekte, die in einer bestimmten Beziehung zueinander stehen.

Es gibt fünf Typen von Beziehungen:

- Service relationship ist eine asymmetrische Beziehung, in der ein MO den user role (Dienstbenutzer), der andere den provider role (Diensterbringer) spielt (Bild 5.7).

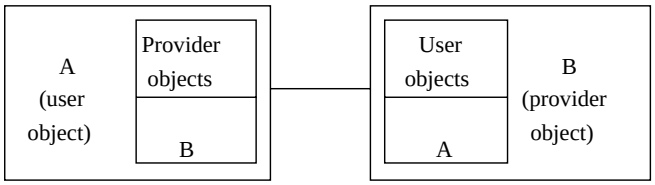


Abbildung 5.7: Service relationship

- Peer relationship ist eine symmetrische Beziehung zwischen zwei gleichartigen MOs.
- Fallback relationship ist eine asymmetrische Beziehung, in der das eine Objekt die "nächste Wahl" für das andere Objekt ist (fallback = zurückgreifen).
- Backup relationship ist eine asymetrische Beziehung, in dem das eine Objekt der Backup des anderen ist.
- Group relationship ist eine Beziehung zwischen einem Besitzer einer Gruppe und einem Mitglied der Gruppe. Sie dient dazu, MOs hinsichtlich einer bestimmten Management-Aufgabe zu gruppieren.

Dienste

Da es für einen Manager notwendig ist, Beziehungen zwischen den MOs zu kennen, gegebenfalls zu ändern und Änderungen aufgrund anderer Einflüsse zu erfahren, werden innerhalb der relationship-management function Dienstprimitive dafür definiert.

Wie bei der state-management function werden die Operationen auf Beziehungsattributen mit Hilfe der bereits definierten pass-through services `PT-GET` und `PT-SET` ausgeführt.

Für die Bekanntmachung der Änderungen einer Beziehung wird ein neuer Dienst definiert, relationship-change-reporting service. Dieser wird mit Hilfe von `M-EVENT-REPORT` erbracht, `event-type` hat dabei den Wert `relationship change`.

5.3.4 Alarm-Reporting Function

Innerhalb der `alarm-reporting function` werden generische Melddungstypen definiert, zusammen mit ihren Parametern und deren Semantik. Diese Meldungstypen charakterisieren Fehler oder abnormale Situationen, die in Netzen auftreten können.

Alarmdefinition

Es werden fünf Typen von Alarmen definiert:

- communications alarm für Fehler in den Komponenten für die Datenübertragung,
- quality of service alarm für Fehler oder Abnahme der Dienstqualitäten eines Objektes,
- processing alarm für Fehler bei der Verarbeitung in einem MO,
- equipment alarm für Komponentenfehler,
- environmental alarm für Fehler in der Umgebung einer Komponente (z.B. bei Ausfall der Klimaanlage).

Für diese Meldungstypen werden Parameter vorgeschrieben, einige davon sind: probableCause, specificProblems, perceivedSeverity, backedUpStatus, etc. (näheres dazu siehe [Sta93g], S. 488).

Dienste

Der Dienst für die Bekanntmachung von auftretenden Alarmen in einem MO heißt alarm-reporting service und verwendet `M-EVENT-REPORT`, wobei `event-type` den Alarmtyp angibt.

5.3.5 Event-Report-Management Function

Die event-report-management function ermöglicht die Steuerung des Meldungsstromes von event reports der MOs, unabhängig von der Definition der MOs. Dafür werden zusätzliche Objekte definiert, die event-forwarding discriminators (EFD). Diese Objekte enthalten unter anderem einen Satz von Bedingungen, die von den Meldungen erfüllt werden müssen, um weitergeleitet zu werden (Filterfunktion). Dabei ist es notwendig, die Meldungen beschreiben zu können, gegebenfals diese Beschreibungen ändern zu können, das Ziel der Meldungen spezifizieren zu können und das Weiterleiten von Meldungen temporär unterbinden und wieder aktivieren zu können. Es ist außerdem notwendig, die Möglichkeit zu haben, alternative Ziele für Meldungen anzugeben, wenn das primäre Ziel nicht erreichbar ist.

Event-Report-Management-Modell

Bild 5.8 zeigt das konzeptionelle Modell für das event-report-management.

MOs generieren die Meldungen, die im lokalen System installierte Instanz event detection and processing empfängt sie und generiert daraus durch Zugabe von Information einen potential event report, der an alle event-forwarding discriminator objects im System verteilt wird. Erfüllt die Meldung die Bedingungen eines der EFDs, so schickt dieser die Meldung an das in ihm spezifizierte Ziel ab.

Dienste

Um das Weiterleiten von events aus einem entfernten Knoten beeinflussen zu können, sind Dienste notwendig, die EFDs erzeugen, löschen und modifizieren können. Erzeugen und Löschen werden mit Hilfe von `PT-CREATE` und `PT-DELETE` erbracht, Modifizieren mit Hilfe von `PT-SET`.

5.3.6 Log-Control Function

Innerhalb der `log-control function` erfolgt die Kontrolle der Speicherung von Informationen über relevante Ereignisse im Netz. Die Funktion unterstützt dabei folgende Aufgaben ([RF91]):

- Setzen und Modifizieren der Auswahlkriterien für zu speichernde Informationen,
- temporäres Anhalten und erneutes Starten der Speicherung,
- Lesen und Löschen der aufgezeichneten Information.

Die Informationen, die gespeichert werden, stammen von der event-report function, von CMIP-PDUs und von Meldungen über interne Ereignisse.

Jeder Informationsspeicher wird als Mo definiert, dem log-object. Jede Information im log-object ist ihrerseits auch ein MO, das log-record, siehe Bild 5.9.

Dienste

Die Dienste der log-control function dienen zum Erzeugen von log-objects (verwenden `PT-CREATE`), zum Löschen von log-objects und log-records (mit Hilfe von `PT-DELETE`), zum Ändern von log-object Attributen, Suspendieren und wieder Fortsetzen der Log-Aktivität (anhand `PT-SET`) und zum Lesen der gespeicherten Informationen, also der log-records (mit Hilfe von `PT-GET`).

5.3.7 Security-Alarm-Reporting Function

Die security-alarm-reporting function dient der Überwachung sicherheitsrelevanter Objekte. ISO definiert Sicherheitsalarme (security alarms) mit Parametern und Semantik und stellt Dienste bereit die EFDs betreffen, welche Sicherheitsmeldungen kontrollieren.

Security Alarms

Es werden fünf Typen von security alarms definiert:

- Integrity violation zeigt an, daß eine potentielle Unterbrechung im Informationsfluß stattgefunden hat (d.h. daß Information illegal modifiziert sein könnte).
- Operational violation zeigt an, daß der verlangte Dienst nicht bereitgestellt werden kann (aufgrund von Nichtvorhandensein, Funktionsstörungen oder einer inkorrekten Dienstanfrage).

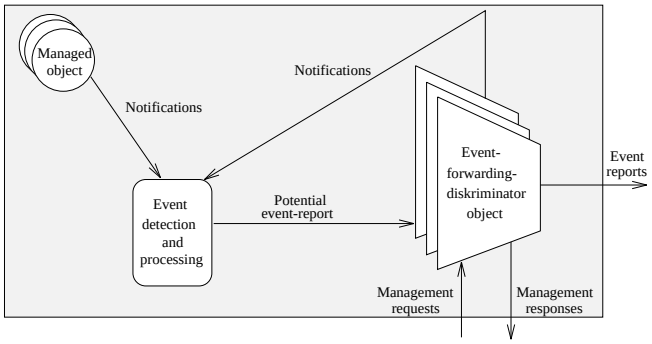


Abbildung 5.8: Modell des event-report-management

- Physical violation zeigt eine Verletzung einer physikalischen Ressource an.
- Security-service or mechanism violation zeigt an, daß ein Sicherheitsproblem entdeckt wurde (Authentifizierungs-Scheitern, unauthorisierter Zugang).
- Time-domain violation zeigt an, daß ein Ereignis außerhalb des ihm gestatteten Zeitraumes erfolgt ist.

Dienste

Der security-alarm-reporting service erlaubt einem user Sicherheitsalarme zu melden. Er wird mit Hilfe von `M-EVENT-REPORT` erbracht, der Parameter `event-type` zeigt den Alarmtyp an.

5.3.8 Security-Audit-Trail Function

Die security-audit-trail function ist eine Erweiterung der log-function. Sie ist dafür verantwortlich, event reports zu spezifizieren, die für die Sicherheitsauswertungen in ein log gespeichert werden.

Die events, die dafür in Frage kommen, können z.B. folgende Ereignisse sein: Verbindungsaufbau, Verbindungsabbau, Verwendung von Sicherheitsmechanismen, Management-Operationen, Buchhaltung über die Benutzung.

Der security-audit-trail service nutzt `M-EVENT-REPORT`, wobei der Parameter `event-type` zwei Werte annehmen kann:

- service report für ein event bezüglich eines Dienstes,
- usage report für die Anzeige eines records mit statistischen Informationen.

5.3.9 Access-Control-Management Function

Die access-control-management function definiert ein Modell für die Kontrolle des Zugriffs auf Informationen und Operationen und spezifiziert MOs und Attribute, um den Zugang zu Ressourcen des Netzes freizugeben oder zu verwehren.

Die Zugangskontrolle (access control) kann verschiedenartig sein, z.B. können manche Benutzer auf Ressourcen Lese- und Schreibzugriff haben, andere nur Lesezugriff oder keinen Zugriff. Eine andere Form der Zugangskontrolle ist die Verhinderung, daß Management-Informationen unauthorisierten Personen bekannt wird oder daß unauthorisierte Personen Zugang zu Management-Operationen haben.

Access-Control Mechanismen

Die access-control-management function unterstützt drei Kategorien von access-control Mechanismen:

- access-control lists,
- capability tickets,
- security labels.

Access-control lists sind Felder von Listen, wobei für jede Ressource (target) eine Liste von Benutzern im weitesten Sinne (initiator) zusammen mit ihren Zugriffsrechten (access rights, z.B. read, write, execute) gespeichert wird.

Access-control list - target1: Initiator1 (read, execute)
Access-control list - target2: Initiator1 (read, write)
Access-control list - target3: Initiator2 (read)

Abbildung 5.10: Access-control list

Die capability tickets enthalten für jeden Benutzer eine Liste der ihm zugänglichen Ressourcen, und der Zugriffsrechte, die dieser Benutzer darauf hat.

Bei den security labels entspricht jeder Ressource eine Klassifizierungsstufe (classification level) und jedem Benutzer eine Unbedenklichkeitsstufe (clearance level). Dabei gibt es dann Zugriffsrechte wie No read up (der Benutzer kann keine Ressourcen lesen, die eine höhere

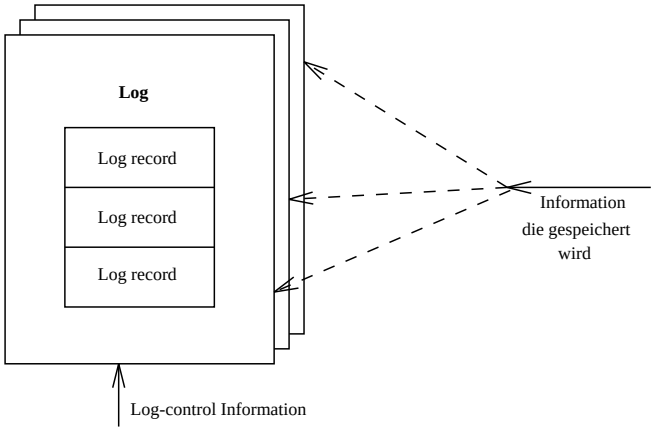


Abbildung 5.9: Log-control Modell

Capability ticket - initiator1

Target1 (read, execute)
Target2 (read, write)

Capability ticket -initiator2

Target3 (read)

Abbildung 5.11: Capability ticket

Sicherheitsstufe haben) oder No write down (der Benutzer kann nicht auf Ressourcen schreiben, die eine niedrigere Sicherheitsstufe haben).

Access-Control Objects

Eines der Ziele der access-control-management function ist die Trennung der Management-Informationen von den Mechanismen, die sie schützen. Daher wird die access-control Information als eine Menge von MOs modelliert, die von den access-control Mechanismen genutzt werden, um die Zugriffskontrolle zu gewährleisten. Es gibt drei Klassen von access-control objects:

- access-control-policy objects,
- targets objects,
- authorized-initiators objects.

Für einen gewissen Sicherheitsbereich gibt es einen access-control-policy object, der die Regeln und access-control Mechanismen des Bereichs beinhaltet. Der Bereich wird durch targets objects beschrieben, wobei jeder targets object ein geschütztes MO identifiziert. Die authorized-initiators objects identifizieren die Zugriffsrechte von Benutzern auf Ressourcen.

5.3.10 Accounting-Meter Function

Die accounting-meter function spezifiziert ein Modell zur Buchhaltung über die Nutzung der Ressourcen und Mechanismen zur Limitierung dieser Nutzung.

Definition der Objekte

Accounting-meter-control object repräsentiert eine Menge von Regeln für die Sammlung von Daten über die Nutzung von Ressourcen. Accounting-meter-data object repräsentiert die Nutzung einer Ressource durch einen Benutzer. Accounting-record object enthält gespeicherte Daten aus einem accounting-meter-data object und aus Meldungen über accounting-meter-data objects.

Dienste

Accounting-meter-action service wird von einem Benutzer der accounting-meter function aufgerufen, um eine bestimmte Aktion einer Partnerinstanz in einem anderen Knoten einzuleiten (wird mit Hilfe von `M-ACTION` erbracht). Accounting-meter-control-notifications service bringt die Informationen über gewisse Aktionen (in Form von Meldungen) aus einem accounting-meter-control object zum gewünschten Ziel (verwendet `M-EVENT-REPORT`). Die Meldungen über die Nutzung von Ressourcen, die von accounting-meter-data objects generiert werden, werden mit Hilfe von accounting-meter-data-record-notification service weitergeleitet (und unter Verwendung von `M-EVENT-REPORT`).

5.3.11 Workload-Monitoring Function

Die workload-monitoring function spezifiziert ein Modell zur Überwachung der leistungsrelevanten Attribute der MOs. Sie definiert MOs, welche Meldungen senden, wenn sich die Werte von Zählern (counters und gauges), die Informationen über die Leistung des Systems enthalten, verändern. Sie dient dazu, rechtzeitig Überlastungen der Ressourcen im Netz zu erkennen.

Es werden dafür metric objects definiert. Diese sind MOs, mit Attributen, die aufgrund von Werten von Attributen aus anderen, überwachten MOs berechnet.

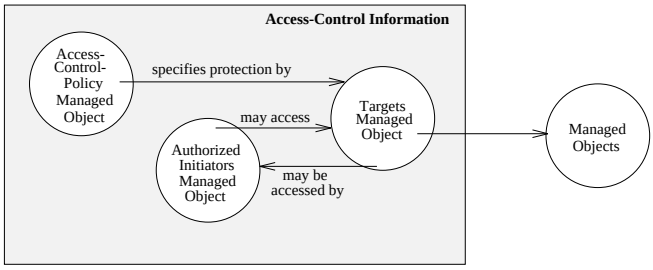


Abbildung 5.12: Beziehung zwischen access-control objects

5.3.12 Test-Management Function

Die test-management function dient der Initiierung und Kontrolle von Tests in entfernten Systemen. Sie spezifiziert Test-Modelle, um Tests auszuführen, die das Ziel haben, die Funktions- oder Leistungsfähigkeit von Ressourcen zu überprüfen.

Test-Modell

Jeder Test umfaßt eine Test-Initiierung durch den Test Conductor über ein Test request, die Ausführung des Tests durch den Test Performer, die Lieferung des Ergebnisses durch ein *Test response* und die Wiederherstellung der normalen Betriebsumgebung.

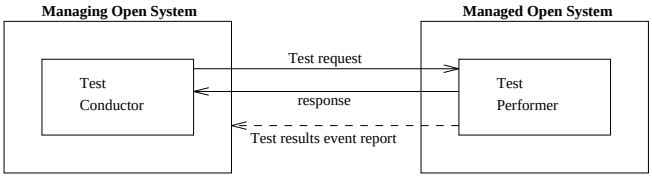


Abbildung 5.13: Generisches Test-Modell

Es werden synchrone und asynchrone Tests unterschieden. Bei den synchronen wartet der Test Conductor auf den Abschluß und die Ergebnisse des Tests, bei dem asynchronen wartet er Test Conductor nicht, sondern die Ergebnisse werden über eine zusätzliche Management-Operation abgefragt.

5.3.13 Summarization Function

Die summarization function dient der Definition von statistischen Maßen und dem Melden von statistischen Informationen. Dabei werden Informationen von anderen MOs abgefragt und in summarization objects plaziert. Diese Informationen werden aus Attributen der MOs, aus metric objects, log records, etc. ... erhalten. Im summarization object wird daraus mittels eines Algorithmus die statistische Information berechnet.

5.4 Literaturverzeichnis

[Sta93e] [Sta93d] [Sta93a] [Sta93g] [RF91]

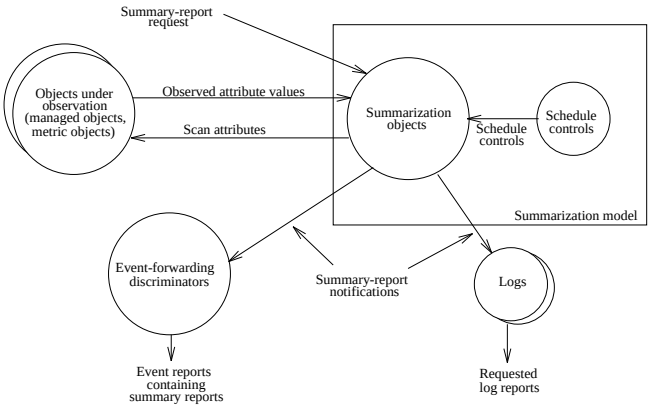


Abbildung 5.14: Modell der OSI-Management-Architektur

Kapitel 6

Hub-centered Management

Frédéric Martin

6.1 Einführung

Das Hub-centered Management ist eine neue Technologie, die das erste Mal in 1992 bei Gelegenheit einer Weltkonferenz über Netzwerke und Telekommunikations (INTEROP FALL 91, 7-11 Oktober 1991, San Jose, California, USA) vorgestellt worden ist.

Diese Ausarbeitung ist mit Hilfe der Folien von der Vorstellungskonferenz dieser Technologie geschrieben worden. Die drei beteiligten Firmen an dieser Vorstellung waren **Hughes LAN Systems**, **Cabletron Systems** und **Synoptics Communications**, die drei der wichtigsten Hersteller für Netzwerks- und Telecommunikationsprodukte sind. Diese Technologie hat seit dieser Zeit einen großen Erfolg gehabt, und das neue Informatikgebäude der Universität Karlsruhe ist schon damit ausgerüstet.

6.1.1 Gesicht der Unternehmensnetzwerke heute

Ein Unternehmensnetzwerk ist heute oft eine Menge von lokalen Netzen, die mit der Hilfe eines Hubs oder eines Rechners, der die Rolle einer Brücke spielt, verbunden werden. Die Tätigkeit von diesen Netzwerken und von jedem dieser verbundenen Rechner wird mit einer Netzwerkmanagementsoftware verwaltet (SNMP, OpenView, usw.). Innerhalb des System Managements werden fünf funktionale Bereiche definiert:

- Fault-Management
- Accounting-Management
- Configuration-Management
- Performance-Managament
- Security-Management

Der Netzwerkmanager arbeitet auf einem besonderen Rechner, der sich irgendwo in dem Netzwerk befinden kann. In den heutigen Netzwerken macht diese überwachende Maschine einen Teil der oben beschriebenen Verwaltungarbeit.

6.1.2 Der Hub

Der Hub ist ein Switchgerät (Schaltergerät), das als flexibler Kommunikationsverteiler dient (Konzentrator, Multiplexer, Pakettechnologie-Integrator, Brücke). Er integriert die verschiedenen benutzten Technologien (Ethernet, Token-ring, FDDI, ISDN, usw.). Aber er besteht heute noch aus einem elektronischen Gerät, d.h. einem Gehäuse, in dem ein oder mehrere Busse liegen und in dem elektronische Karten eingesteckt werden. Er enthält keine oder wenig Software.

6.1.3 Welchen Problemen begegnet man heute ?

- Das Netzwerk wird nicht als ein Objekt betrachtet. D.h., daß es mehr oder weniger durchsichtig und passiv ist, und daß niemand für seine Dienstqualität verantwortlich ist. Jede Operation wird durch die Netzwerkmanagementsstation gemacht; man kann keine Frage direkt dem Netzwerk stellen.
- Es gibt zu viel Verkehr der Verwaltungsdaten, weil die Netzwerkmanagementstation nicht an einer zentralen Stelle ist. Deshalb laufen die Verwaltungsdaten (z.B. Verfügbarkeits- oder Statistikdaten) einen zu langen Weg durch das Netzwerk. Manchmal können diese Daten mehr als 25 Prozent des gesamten Datenverkehrs ausmachen.
- Das Netzwerk ist empfindlich, weil, wenn es ein Problem auf Teil des Netzwerks, wo die Netzwerkmanagementstation liegt, gibt, die Verwaltung des Netzwerk nicht mehr gesichert ist. Außerdem passieren die selben Schwierigkeiten, wenn die Verbindungen zwischen den verschiedenen Netzwerkteilen beschädigt oder zerstört sind.

6.1.4 Die Trends im Netzwerkmanagement

- Netzwerke vergrößern sich, aber ihre Segmente werden kleiner und kleiner. Eine Konsequenz ist, daß die Unterschiede zwischen LAN (Local Area Network) und WAN (Wide Area Network) zu verschwinden neigen.

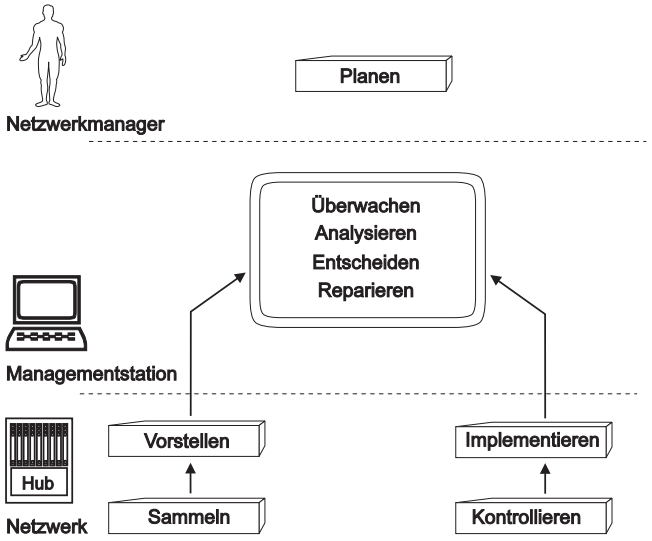


Abbildung 6.1: Typische Netzwerkmanagementorganisation

- Netzwerke transportieren mehr und mehr Daten für Anwendungen, deren Anforderungen immer größer sind (verteilte Datenbanken, multimediale Anwendungen, distributed processing, Telekonferenz und kooperative Arbeit). Mehr Leistungsfähigkeit wird gefordert.
- Anforderung eines echten ausfallsicheren Dienstes (Fault-tolerant networks for fault-tolerant applications), dessen Grundfunktionen die Entdeckung, die Aufspürung, die Identifizierung, die Behandlung, die Vorbeugung vor Fehlern sind. Dieser Dienst kann eventuell alternative Diensten versorgen und seine Wiederherstellung durch Reparatur sichern.

6.2 Die Hub-centered Management Lösung

6.2.1 Die Rolle des Hubs wird umdefiniert

Das Netzwerk muß, um die obenbeschriebenen Probleme zu lösen, durch eine Verbesserung des Hubs intelligent werden. Der Hub wird als das Herz des Netzwerks betrachtet, aber muß auch die Rolle eines Gehirnes spielen, um sich allein zu verwalten und zu überwachen.

6.2.2 Implementierung der Intelligenz in einem Hub

Physikalisch besteht die Implementierung aus einer oder mehreren elektronischen Karten, jede mit einem Intel 80960 Prozessor ausgerüstet. Es gibt auch bis 12MB DRAM für die Managementdaten, Flash EPROM statt eine normalen Festplatte, 4MB sehr schnelle Packet DRAM, um die Daten zu kontrollieren und zu senden.

Auf der logischen Ebene wird ein objekt-orientiertes Betriebsystem installiert, um eine echte Kommunikation zwischen den verschiedenen Objekten zu sichern. Die Daten werden durch die Protokolle IP (Netzwerkniveau) und durch UDP (Transport-niveau) übertragen. UDP, User Datagram Protocol, ist ein Dienst, der parallel zu TCP existiert, und der eine transaktions-orientierte Datenblockübertragung ohne Bestätigung erlaubt. Dazu kommen die Komponenten, die normalerweise auf der Netzwerkmanagementstation installiert werden. D.h. das Netzwerkmanagementprotokoll SNMP (Simple Network Management Protocol), und die MIB (Management Information Base, das Gedächtnis des Netzwerks), die in diesem Fall Hub-MIB genannt wird.

Diese Implementierung der Intelligenz in dem Hub ist eine notwendige Bedingung, um die Managementstation zu ersetzen.

6.2.3 Was muß der neue intelligente Hub machen?

Einen aktualisierten Blick des Netzwerkzustandes behalten

In seinem Speicher behält der Hub Informationen über den Netzwerkzustand. Zwei Datenarten werden hier gesammelt:

- die Konfigurationsdaten, die mehr oder weniger statisch sind, stellen Angaben über das verwaltete System (z.B. Systemname, angeschlossenen Netzwerke) dar.
- die Zustandsdaten sind dynamische Informationen über die Verfügbarkeit der verschiedenen Netzwerkteile, die Fehlerrisiken und den Datenverkehr.

Integrierte Zugriffskontrolle

Der Hub befindet sich physikalisch am Eingang eines lokalen Netzwerks, das heißt auf dem Weg der Daten, die von außerhalb kommen. Deshalb enthält er die Benutzer- und Rechteliste und soll die Zugriffsrechte überwachen.

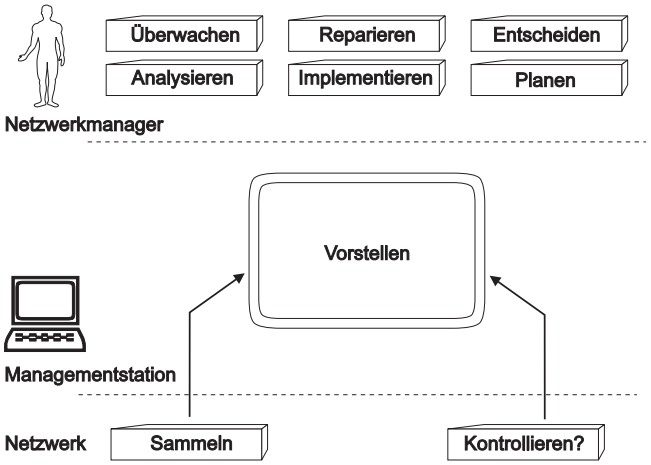


Abbildung 6.2: Verteilte Netzwerkmanagementorganisation

Fehlerlokalisierung und -verarbeitung

Der Hub prüft regelmäßig die Verfügbarkeit der verschiedenen Ressourcen. Er unterhält also für jedes Objekt (d.h. Rechner, Verbindung, Netzwerkteil oder Ressource) Verfügbarkeits- und Benutzungsstatistiken. Falls sich ein Fehler erreignet, z.B. der Rechner ist ausgeschaltet oder es gibt zu viele Fehler auf einem Netzwerkteil, ist der Hub fähig, wenn nötig, die entsprechenden Ressourcen durch ihre Sicherheitsressourcen zu ersetzen.

„Network recovery“ Hilfe

Falls es einen großen Unfall gibt, zum Beispiel ist das Netzwerk total oder teilweise zerstört, ist der Hub immer der erste, der wieder aktiv wird. Er ist fähig den ursprünglichen Zustand des Netzwerkes nach einem Unfall, zwar nicht identisch, aber sehr genau, schnell wieder herzustellen.

6.2.4 Vorteile dieser neuen Technologie

Leistungfähigkeit, Zuverlässigkeit und Erweiterungfähigkeit sind die drei Bereiche, wo diese neue Technologie die größten Fortschritte gemacht hat:

1. Leistungfähigkeit: Zwei wichtigen Faktoren tragen zur Verminderung des Verwaltungsverkehr bei:
   - Zuerst werden die Daten, die das Netzwerk betreffen, vor Ort vorbereitet, d.h. Statistikdaten, Benutzer- und Zugriffsrechteliste, Hub-MIB.
   - Zweitens, falls das Netzwerk groß genug ist, wird die Netzwerkintelligenz durch die Implementierung von mehreren Hubs verteilt.
2. Zuverlässigkeit
   - Netzwerkteile können Probleme haben, die Netzwerkmanagementstation kann abstürzen. In diesen Fällen kann der Hub entscheiden, was zu tun ist, um das Netzwerk am besten funktionsfähig zu halten. Wenn das Netzwerk die Verantwortung über sein eigenes Funktionieren übernimmt, dann erreicht man eine erhöhte Zuverlässigkeit.
3. Erweiterungfähigkeit
   - Es ist einfacher mit dieser Organisation die Topologie des Netzwerks zu ändern oder neue Technologien einzuführen, weil der Hub relativ unabhängig von der Netzwerktechnologie ist.

6.3 Das neue Netzwerkmanagement

6.3.1 Neue Ziele des Netzwerkmanagers

In der typischen Netzwerkmanagementorganisation sammelt die Netzwerkmanagementstation die verschiedenen notwendigen Informationen, die das Funktionieren des Netzwerks betreffen, und stellt die Ergebnisse dar. Die Handlung des Netzwerkmanagers setzt sich aus fünf Aufgaben zusammen:

- Die Überwachung von dem guten Funktionieren des Netzwerks.
- Die Analyse der Ergebnisse, die von der Managementstation vorgestellt werden (Statistiken, Fehler, ...) und die Entdeckung von eventuellen Fehlern.
- Die täglichen oder außergewöhnlichen Wiederherstellungen und Fehlerlösungen.
- Die Implementierung von Ersatzsoftware, falls sich die Benutzungsbedingungen des Netzwerkes aus irgendwelchen Grund ändern, und die Implementierung von neuen Versionen der schon existierenden Programmen.
- Die Planung, die das Werden des Netzwerks betrifft.

In der verteilten Netzwerkmanagementorganisation existiert noch die Netzwerkmanagementstation, aber sie

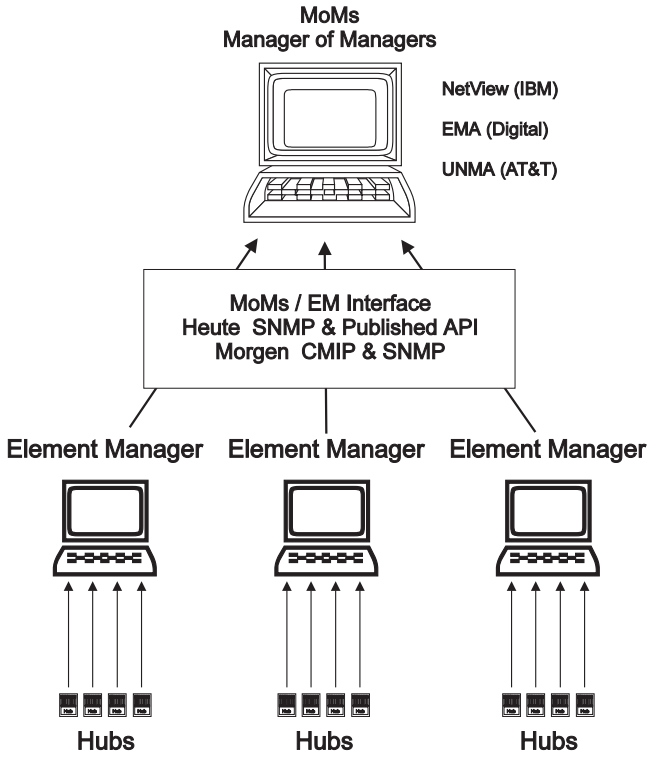


Abbildung 6.3: Hierarchisches Netzwerkmanagements

spielt nur die Rolle eines Interface zwischen dem intelligenten Netzwerk und dem Netzwerkmanager. Mit diesem Modell findet eine große Menge der vorangehenden Handlungen in dem Hub statt. Der Manager behält nur die Aufgabe der Planung.

6.3.2 Die Hierarchie der Netzwerkintelligenz

Ein Trend der Informatik ist, wegen des immer geringeren Preis von der Mikroprozessoren, daß man eine mehr und mehr große Anzahl von Mikroprozessoren jedem Benutzer zur Verfügung stellt. In einer solchen verteilten Organisation, insbesondere wenn sie eine bestimmte Mindestgröße hat, gibt es auch Bedürfnisse einer Hierarchie. Unter SNMP, man kann zum Beispiel ein strukturiertes Netzwerk planen, das verschiedene Netzwerke umfaßt. Jedes Netzwerk wird mit einem Hub kontrolliert, der die Daten verarbeitet und filtert, bevor sie zu der Netzwerkmanagementstation gesendet wird.

Für besonders große Unternehmensnetzwerke kann man eine echte hierarchische Struktur konstruieren. Mehrere Netzwerke werden von einem MoMs (Manager of Managers) kontrolliert. Die Netzwerkmanagementstationen jedes Netzwerkes werden Element Manager genannt. Diese Architektur stützt sich auf die Benutzung von SNMP und CMIP (Common Management Information Protocol, ein Protokoll auf der Applikationsebene, um Managementinformationen auszutauschen).

Verschiedene Managementsoftware existiert schon für solche Netzwerkorganisation:

- NetView (**IBM**)
- EMA (Enterprise Management Architecture) (**DIGITAL**)
- UNMA (Unified Management Architecture) (**AT&T**)

6.4 Schlußfolgerung

Was die Konstrukteure dieser Technologie machen wollten, ist, die verschiedenen Komponenten eines Netzwerkes zu identifizieren und zu isolieren, sie als Objekte zu betrachten, um ihre Rolle, aufgrund der Dienste, die sie leisten, genau zu bestimmen.

6.5 Literaturverzeichnis

[JJMD92]

Kapitel 7

Eine generische Netzwerkmanagementarchitektur

Thomas Ehrenberg

7.1 Einführung

Die bisher behandelten Protokolle (SNMP, OSI, ...) beruhen auf zum Teil völlig verschiedenen Ansätzen und Architekturen. Je nach Zielsetzung und Anforderung an das Netz (betreffend z.B. angebotene Dienste, Sicherheit, Verfügbarkeit, Erweiterbarkeit des Netzes), sowie an dessen Verwaltung (also an das Netzwerkmanagement) ergeben sich verschiedene optimale Managementarchitekturen. Durch diese Verschiedenartigkeit entstand eine starke *Heterogenität* der einzelnen Managementarchitekturen.

Zur Verwaltung von Netzen ist nun eine Grundvoraussetzung, daß alle zu verwaltenden Geräte in diesem Netzwerk das jeweils gewählte Protokoll unterstützen. Dies kann aufgrund der gewachsenen Struktur in vielen Betrieben heute nicht mehr unbedingt vorausgesetzt werden:

- die zunehmende elektronische Kommunikation erfordert mit steigendem Vernetzungsgrad mächtigere, schnellere Protokolle. Daraus resultiert die Notwendigkeit der ständigen Weiterentwicklung bestehender Protokolle bzw. einer völligen Neukonzeption.
- bei der Expansion einer Firma reicht das bisherige Netz (-Protokoll) nicht mehr aus.
- bei der Fusion zweier Firmen mit verschiedenen Netzarchitekturen müssen beide Netze in ein Netzsystem integriert werden.

Es wird also nötig, verschiedene Architekturen und Protokolle in einem Netz zu vereinen und zusammen zu verwalten.

Bisherige Integrationsmöglichkeiten sind noch sehr unbefriedigend und mit vielen Nachteilen verbunden (auf die existierenden Ansätze und ihre Eigenschaften wird in den Abschnitten 7.2.2 und 7.2.3 näher eingegangen). In [Sei94], auf dem die vorliegende Ausarbeitung beruht, wird nun mit Hilfe eines *generischen Ansatzes* versucht, die bisher erkannten Nachteile der bestehenden Lösungen zu vermeiden und die unterschiedlichen Architekturen in einem umfassenden Netzwerkmanagement zu integrieren.

7.2 Heterogenes Management

Dieses Kapitel untersucht zunächst, welche Möglichkeiten bei der Verwendung verschiedener Managementstandards bestehen, ein solches heterogenes Netz zu verwalten.

7.2.1 Kriterien für Integrationsansätze

Zur Bewertung der Integrationsmöglichkeiten verschiedener Managementstandards werden folgende Kriterien herangezogen:

Transparenz gegenüber dem verwendeten Standard:
Wieviel muß eine Anwendung vom reell verwendeten Standard wissen ?

- **keine Transparenz** : die Anwendung muß auf den verwendeten Standard zugeschnitten sein.
- **Protokolltransparenz** : das verwendete Protokoll hat keine Auswirkung auf die Anwendung.
- **Modelltransparenz** : es besteht keine Festlegung bezüglich des Modells der Managementinformation (z.B. MIB-Aufbau).
- **völlige Transparenz** : sie vereinigt Protokoll- und Modelltransparenz.

Verdeckung unterschiedlicher Monitortechnologien:
Die Anwendung soll nicht von der Art der verwendeten Monitortechnologie (Hardware-, Software- oder Hybrid-Monitor) abhängen.

Unterstützung unterschiedlicher Zielrichtungen bei den Managementanwendungen:
Mögliche Zielrichtungen sind beispielsweise Dienstemanagement, Konfigurationsmanagement, Sicherheitsmanagement, Abrechnungsmanagement.

Erweiterbarkeit / Flexibilität :
Durch die rapide Entwicklung im Bereich Telekommunikation ist gerade dieser Punkt sehr wichtig geworden (im Hinblick z.B. auf Agenten neuer Geräte, die die alten Protokolle nicht mehr unterstützen).

Ressourcenanforderungen :
Managementtätigkeiten sollten die Ressourcen im Netz möglichst wenig belasten.

Anzahl der Knoten mit Managementaufgaben :
Ziel ist es, möglichst wenige Knoten mit Managementaufgaben zu belegen. Außerdem sind Informationen, die an einer Stelle zusammenlaufen auch leichter handzuhaben.

Fehlertoleranz :
Fehler sollten weitestgehend keine Auswirkungen auf die Managementanwendung haben. Beim Ausfall des Knotens, auf dem die Anwendung läuft, muß diese möglichst einfach auf einem anderen Netzknoten gestartet werden können. Der Ausfall eines Agenten hingegen führt in allen Fällen zum gleichen Ergebnis, nämlich daß die von ihm verwaltete Information nicht mehr zugreifbar ist.

Ein Management wird als *umfassend* bezeichnet, wenn es die obigen Kriterien erfüllen kann.

7.2.2 Ansätze zur Integration von Managementarchitekturen

Im folgenden werden fünf Ansätze vorgestellt, die es erlauben, ein heterogenes Netz zu verwalten:

1. **Insellösung**
   Die einzelnen Komponenten (-gruppen) werden jeweils von einem passenden Manager verwaltet. Dies führt zu einer größeren Anzahl nicht miteinander kooperierender Manager, die jeweils ihre eigenen Managementanwendungen haben.

   Eigenschaften Da jeder Manager seine eigenen Anwendungen hat, ist keine Transparenz gegenüber dem verwendeten Protokoll notwendig. Bei der Erweiterung um Agenten, die ein bisher noch nicht unterstütztes Protokoll erfordern, entsteht eine neue Insel, d.h. es muß ein neuer Manager mit eigenen Anwendungen, die auf diesem Protokoll aufbauen, realisiert werden.

2. **Insellösung mit Koordinator**
   Eine Erweiterung der Insellösung besteht darin, daß ein koordinierender Manager über ein spezielles *Manager-to-Manager–Protokoll* mit den einzelnen Inselmanagern kommuniziert. Dadurch ist eine beschränkte Einflußnahme auf die Funktionalität der so entstandenen Submanager möglich, eine Kooperation der Submanager untereinander ist aber weiterhin ausgeschlossen. Der Koordinator hat auch keinen direkten Zugriff auf den Agent, er kann Informationen nur über den Submanager anfordern.

   Eigenschaften Anwendungen, die auf dem Koordinator laufen, sind transparent gegenüber den Protokollen der Submanager, Anwendungen auf den Submanagern müssen aber weiterhin auf das dort vorhandene Protokoll zugeschnitten sein. Ebenso muß für neue Agenten mit neuem Protokoll auch ein neuer Submanager realisiert werden. Vorteil zum vorherigen Ansatz ist, daß dann auch auf diesen neuen Submanager durch das *Manager-to-Manager–Protokoll* zugegriffen werden kann. Nachteilig wirkt sich dagegen die zusätzliche Instanz für den kooperierenden Manager aus.

3. **Multilinguale Agenten**
   Die Agenten werden mit mehr als einem Protokoll ausgestattet (dieser Ausbau der Agenten muß vom Anbieter erbracht werden).

   Eigenschaften Die Anwendung muß nur noch das eine, von allen Agenten verstandene Protokoll unterstützen. Dadurch müssen neue Agenten aber immer zu diesem Protokoll kompatibel bleiben. Die geforderte Mehrfachausstattung macht die Agenten ziemlich umfangreich (z.B. Unterstützung des OSI–Protokollturms bis Schicht sieben und des TCP/IP–Stacks bis Schicht vier).

4. **Multilingualer Manager**
   In Umkehrung zum vorherigen Ansatz unterstützt hier der Manager mehrere Protokolle. Dieser Ansatz entspricht der Möglichkeit der Insellösung, mit dem Unterschied, daß die einzelnen Manager zu einem einzigen verschmolzen werden.

   Eigenschaften Die Anwendungen können nicht transparent bezüglich des Protokolls sein, da sie selbst dafür verantwortlich sind, mit welchem Protokoll ein Agent zu erreichen ist. Bei Eingliederung eines neuen Agenten muß der Manager um dieses Protokoll erweitert werden.

5. **Integrierendes Management**
   Die bisher vorgestellten Ansätze haben die gestellten Forderungen nur zum Teil erfüllen können. Bislang wurden entweder für die Unterbereiche getrennte Anwendungen eingesetzt, die nicht mit anderen kooperierten, oder die Anwendungen waren nicht transparent gegenüber dem eingesetzten Protokoll. Um letzteres zu erreichen, muß eine Schicht eingeführt werden, die das von der Anwendung unterstützte Protokoll transparent in das vom Agenten unterstützte umsetzt. Dabei muß auch das zugehörige Modell, also die Darstellung der Managementinformation, abgebildet werden.

   Die Realisierung dieser *Konvergenzschicht* kann auf zwei Arten geschehen:

   (a) **Gateway–Ansatz:**
   Die einzelnen zu verwaltenden Komponenten

werden jeweils über ein Gateway mit dem Manager verbunden. Der Unterschied zur Insellösung besteht darin, daß die Gateways keine Managementanwendungen realisieren, sondern nur für die Protokollumsetzung zuständig sind. Die Ressourcenanforderungen sind hier wesentlich geringer als bei einem Manager.

(b) **Plattform–Ansatz:**
Die Abbildung wird direkt innerhalb des Managers realisiert, so daß die Anwendung ihre Anfrage an einen Agenten über eine Plattform schickt, welche die Anfrage entsprechend dem vom Agenten unterstützten Protokoll umwandelt. Bei der Antwort erfolgt die Umwandlung ebenfalls in der Plattform. Dadurch können die Ressourcen für die Gateway–Lösung eingespart werden, der Aufwand muß allein beim Manager getrieben werden.

7.2.3 Beurteilung der Ansätze

In Tabelle 7.1 werden die vorgestellten Verfahren anhand der eingeführten Kriterien verglichen.

- *Transparenz gegenüber dem verwendeten Standard:*
  Unterschiedliche Standards können bei der Insellösung und dem multilingualen Manager nicht verdeckt werden, da dort die Anwendungen auf den Standard zugeschnitten sein müssen. Die Insellösung mit Koordinator verdeckt die Unterschiede nur im Koordinator.

- *Verdeckung unterschiedlicher Monitortechnologien:*
  Hierfür sind die Agenten verantwortlich, so daß sich nur im Ansatz mit multilingualen Agenten Probleme ergeben können, da sonst die Agenten unangetastet bleiben.

- *Unterstützung unterschiedlicher Zielrichtungen:*
  Sowohl bei der Insellösung als auch im multilingualen Manager können je nach Zielrichtung neue Inseln entstehen. Die Insellösung mit Koordinator erfüllt dieses Kriterium wieder nur im Koordinator.

- *Erweiterbarkeit / Flexibilität:*
  Bei den beiden Varianten der Insellösung muß zur Erweiterung jeweils ein neuer Manager realisiert werden. Bei dem multilingualen Manager ist zwar das Einfügen eines neuen Protokolls in den Manager relativ einfach zu realisieren, allerdings müssen dann neue Anwendungen dafür erstellt werden.

- *Ressourcenanforderungen:*
  In beiden Varianten der Insellösung muß für jeden unterstützten Standard eine eigene vollständige Managementinsel realisiert werden. Beim multilingualen Agenten sind die Anforderungen an die Agenten aufgrund deren erweiterten Fähigkeiten hoch. Der Gateway–Ansatz benötigt zusätzliche Ressourcen zur Realisierung der Gateways, die allerdings nicht so groß sind wie bei der Insellösung mit Koordinator.

- *Anzahl der Knoten:*
  Bei multilingualen Agenten und Managern, sowie beim Plattform–Ansatz kann die Anzahl der Knoten mit Managementaufgaben auf einen begrenzt werden. Bei den anderen Ansätzen müssen zusätzliche Knoten bereitgestellt werden.

- *Fehlertoleranz:*
  Die fehlende Toleranz gegenüber dem Ausfall des Knotens, auf dem der Manager realisiert ist, ist ein Hauptschwachpunkt aller vorgestellten Ansätze. Eine Lösung wäre ein *Stand-by-Manager* auf einer anderen Komponente. Auf der anderen Seite könnte eine Mischform des Gateway– oder Plattform–Ansatzes mit der Insellösung mit Koordinator, also eine Managerhierarchie, zumindest garantieren, daß einzelne Teilbereiche weiter verwaltet werden können, auch wenn der übergeordnete Manager ausfällt.

7.3 Eine generische Managementarchitektur

Für diese Architektur wurde der Plattform–Ansatz gewählt. Er bietet eine völlige Transparenz gegenüber den verwendeten Standards, wobei auch unterschiedliche Zielrichtungen unterstützt und verschiedene Monitorarchitekturen integriert werden können. Er ist leicht erweiterbar und flexibel hinsichtlich neuer Standards. Die Anforderungen an Ressourcen für das Management sind relativ niedrig. Außerdem kann mit einer Managementplattform auch hierarchisches Management ermöglicht werden.

7.3.1 Ein allgemeines Management–API (AM-API)

Um die unterschiedlichen Zielrichtungen der Managementanwendungen auf einen Nenner zu bringen, muß der kommunikationsspezifische Teil vom anwendungsspezifischen getrennt werden. Dadurch können Anwendungen unabhängig vom darunterliegenden Kommunikationsmechanismus, der den Zugriff auf die Managementinformationen bietet, definiert werden.

Die Schnittstelle zum Manager wird durch das *Application Programming Interface* (API) definiert (agentenseitig gibt es kein API, da hier die Schnittstelle durch das verwendete Protokoll gegeben ist). Um ein umfassend integrierendes (*allgemeines*) Management zu ermöglichen, muß das vom Manager zur Verfügung gestellte API auch unabhängig vom jeweils verwendeten Protokoll sein.

Zwischen Manager und Agent kommt es zu den folgenden Kommunikationstypen, die in einem solchen AM-API als Grundmenge der möglichen Operationen vorhanden sein müssen:

– **Attributabfrage** : Mit dieser Operation wird die Abfrage eines spezifizierten Attributs einer *Managed Object Instanz* (MOI) veranlaßt. Der Initiator

	Insellösung	Insellösung mit Koordinator	Multilinguale Agenten	Multilinguale Manager	Gateway-Ansatz	Plattform-ansatz
Unterstützung unterschiedlicher Zielrichtungen bei den Managementaufgabe	nicht einheitlich	einheitlich nur im Koordinator	einheitlich	nicht einheitlich	einheitlich	einheitlich
Verdeckung unterschiedlicher Monitor-Architekturen	durch Agenten	durch Agenten	aufwendig durch Integration im Agenten	durch Agenten	durch Agenten	durch Agenten
Transparenz gegenüber unterschiedlichen Managementstandards	keine	nur im Koordinator	völlige	keine	völlige	völlige
Erweiterbarkeit und Flexibilität hinsichtlich neuer Managementstandards	sehr schlecht	sehr schlecht	nicht relevant	schlecht	gut	gut
Anforderungen an Ressourcen im zu verwaltenden Kommunikationsnetz	hoch	hoch	hoch in Agenten, sonst niedrig	niedrig	mittel	niedrig
Anzahl der Knoten mit Managementaufgaben bei n unterschiedlichen Managementstandards	$\geq n$	$\geq n+1$	≥ 1	≥ 1	$\geq n+1$	≥ 1
Auswirkungen des Ausfalls eines Knotens mit Managementaufgaben	Insel nicht verwaltbar	Insel/Gesamtnetz nicht verwaltbar	Gesamtnetz nicht verwaltbar	Gesamtnetz nicht verwaltbar	Insel/Gesamtnetz nicht verwaltbar	Gesamtnetz nicht verwaltbar

Tabelle 7.1: Zusammenfassender Überblick

ist der Manager, die Anfrage wird bestätigt (die Antwort enthält das Ergebnis).

– **Attributänderung** : Der Manager kann damit den Wert eines Attributs einer MOI auf einen bestimmten Wert setzen. Die Operation wird bestätigt (Erfolgsmeldung).

– **Operationsaufruf** : Mit Hilfe dieses Typs kann der Manager eine beliebige, von einer MOI angebotene Operation anstoßen. Der Aufruf wird bestätigt.

– **Ereignismeldung** : Über diese Operation wird ein Manager von einem Agenten über eine Ausnahmesituation benachrichtigt.

Um bereits existierende Anwendungen, die direkt auf die vom Managementprotokoll erbrachten Dienste zugreifen, nicht ändern zu müssen, werden *Bypass–Interfaces* zugelassen, um das AM-API umgehen zu können.

7.3.2 Integration der Monitortechniken

Die verschiedenen Ansätze bei Monitoren (Hardware-, Software- und Hybrid-Monitore) werden durch das Konzept von Agent und Managed Objects (MO's) kompensiert, so daß die Heterogenität für das Netzwerkmanagement verdeckt werden kann. Daher soll dieser Integrationspunkt hier nicht weiter ausgeführt werden. Von Interesse ist nur, wie die Anbindung von MO's bzw. deren Instanzen an die durch Monitore erfaßte Information geschieht.

Dazu werden in [KP93] drei verschiedene Techniken aufgeführt:

– **upon external request :** Sobald bei einem Agenten die Anfrage an eine MOI ankommt, wird der gewünschte Wert vom Monitor bestimmt. Kommunikation zwischen Monitor und MOI findet also nur statt, wenn sie wirklich benötigt wird. Sie kann dann aber zeitaufwendig sein, wenn der Monitor dafür viele Daten beschaffen muß.

– **cache-ahead–Prinzip :** Der Inhalt der MOI stellt eine *"shadow-copy"* der Information des Monitors dar, die in bestimmten Zeitabständen auf den aktuellen Stand gebracht wird. Man kann zwar nicht garantieren, daß die Information in der MOI aktuell ist, die notwendige Aktualität kann aber durch eine geeignete Festlegung des Zeitintervalls erreicht werden.

– **event-driven :** Die bisherigen Methoden sind nur geeignet, wenn die Abfragen vom Manager initiiert werden. Meldungen vom Agenten müssen ereignisgesteuert erfolgen. Dabei aktualisiert der Agent die MOI und veranlaßt dann eine Meldung an den Manager. Dafür muß dem Monitor allerdings bekannt sein, unter welchen Bedingungen das Aussenden einer Meldung geschehen soll.

Generell muß also für jede MOI die Anbindung an den sie mit Informationen versorgenden Monitor gemäß der jeweils gewünschten Charakteristika ausgewählt werden.

7.3.3 Integration der Managed Objects

Bei den verschiedenen Standards zur Beschreibung der MO's und ihrer Instanzen treten Unterschiede hinsichtlich der Benennung, der Definition der Instanzen (dynamisch oder statisch) und der Anzahl der Attribute je MO auf. Mit Hilfe eines *generischen MO's*, das zusätzliches Wissen darüber enthält, wie die in ihm enthaltene Information auch in anderen, heterogenen *MIB's* (Management Information Bases) abgefragt werden kann, können diese unterschiedlichen Beschreibungen integriert werden.

Darstellungsunabhängige Beschreibung eines Managed Object

Ein Managed Object wird nach [Sei94] als 5-Tupel *(Name, Zugriffsauthentisierung, Attributliste, Methodenliste, Meldungsliste)* definiert, wobei gelten soll:

- **Name** = *ASN.1 Octet String*
  Name des MO bzw. dessen Instanz.

- **Zugriffsauthentisierung** = *definitionsabhängig*
  Berechtigungsnachweis zum Zugriff auf das MO.

- **Attributliste** = $\{att_1, \ldots, att_l\}$
  Menge der Attribute des MO, wobei att_i selbst wieder ein 3-Tupel mit den Komponenten *(attributname, attributwert, zugriffsart)* ist:

  - **attributname** = *ASN.1 Octet String*
    Name des Attributs.
  - **attributwert** = *definitionsabhängig*
    Wertebereich bei einem MO bzw. konkreter Wert bei einer MOI.
  - **zugriffsart in** {*read-only, read-write, write-only, not-accessible*}
    Definition des Zugriffs auf das Attribut.

- **Operationsliste** = $\{method_1, \ldots, method_m\}$
  Menge der Operationen, die das MO bzw. dessen Instanz einem Manager zur Verfügung stellt.

- **Meldungsliste** = $\{report_1, \ldots, report_n\}$
  Menge der Meldungen, die für das MO definiert sind.

Die Schreibweisen für den Zugriff werden wie üblich eingeführt:

- $MO.att_i$ bezeichnet das Attribut att_i des MO.
- $MO.method_j(par_1, \ldots, par_n)$ bezeichnet den Aufruf der Operation $method_j$ des MO mit den Parametern $par_1, \ldots, par_n$.
- $MO.report_k(par_1, \ldots, par_m)$ bezeichnet das Eintreffen der Meldung $report_k$ des MO mit den Parametern $par_1, \ldots, par_m$ beim Manager.

Im folgenden wird an zwei Beispielen gezeigt, wie sich ein MO in das vorgestellte Schema einpassen läßt:

Beispiel 1 Die OSI–MIB enthält das Managed Object *bandwithBalancingProcess*. Dieses Object wird definiert als:

Managed Object
Name: *bandwithBalancingProcess (bBP)*
Zugriffsauthentisierung: ...
Attributliste
- attributname : *bBPPackage_bandwithBalancingID*
  - attributwert : *Integer*
  - zugriffsart : *read-only*
- attributname : *bBPPackage_bwbBusIdentifier*
  - attributwert : *Integer*
  - zugriffsart : *read-only*
- attributname : *bBPPackage_bwbModulus*
  - attributwert : *Integer*
  - zugriffsart : *read-write*
- attributname : *bwbActivePackage_bwb_Counter*
  - attributwert : *Counter*
  - zugriffsart : *read-only*

Beispiel 2 Für das Managed Object *bmod* für eine von SNMP verwendete MIB sieht die darstellungsunabhängige Beschreibung wie folgt aus:

Managed Object
Name: *bmod*
Zugriffsauthentisierung: ...
Attributliste
- attributname : *bmod*
  - attributwert : *Integer*
  - zugriffsart : *read-write*

Hier ist auch einer der wichtigsten Unterschiede zwischen den beiden Definitionen für MO's festzustellen: während die OSI-Definition mehrere Attribute zuläßt, hat ein MO einer SNMP–MIB nur ein Attribut, so daß Name des MO und Attributname gleich sind.

Meta-Managed Object

Durch die im vorigen Abschnitt eingeführte Beschreibung eines Managed Objects ist beim Zugriff auf ein MO das zugrundeliegende Modell transparent verdeckt. Ein Nachteil wurde dadurch aber noch nicht behoben:

Die gleiche Information kann in mehreren MO's gefunden werden, so daß die Anwendung wissen muß, welche MO's bei einem Agenten verfügbar sind, um an die gewünschte Information zu gelangen.

Bsp.: Bei den obigen Beispielen ist in den Attributen *bmod.bmod* und *bandwithBalancingProcess.bBPPackage_bwbModulus* die gleiche Information abgespeichert. Ersteres Attribut kann in einem SNMP–Agenten ermittelt werden, das zweite Attribut existiert in der MIB eines CMIP–Agenten. Daher muß die Anwendung, wenn sie von einem Agenten eine Information erhalten will, entweder

zwei Anfragen stellen, von denen eine mit Sicherheit fehlschlägt, oder wissen, welche Darstellung der gewünschten Information vom Agenten unterstützt wird.

Zur Lösung des Problems wird die formale Beschreibung der Attribute aus Abschnitt 7.3.3 erweitert:

- Das 3-Tupel *(attributname, attributwert, zugriffsart)* zur Beschreibung eines Attributs wird durch die Komponente

$$\begin{aligned} &\textit{Attributskorrelationsliste} \\ &= \{attr_{\mathrm{korr}_1}, \ldots, attr_{korr_k}\} \end{aligned}$$

zu einem 4-Tupel erweitert.
$Attr_{korr_i}$ ist dabei ein Tupel *(protokoll, abfragedienst, a-get-korrelation, änderungsdienst, a-set-korrelation)*:

  - **protokoll in** {*CMIP, SNMP, SNMPv2,...*}
    Hier wird für das korrelierende MO das betreffende Protokoll abgelegt.
  - **abfragedienst** = *protokollabhängig*
    Es kann mitunter verschiedene Möglichkeiten geben, die im Attribut gespeicherte Information aus einem MO einer anderen Darstellung zu gewinnen. So kann neben dem Get-Dienst bei SNMPv2 auch der GetBulk-Dienst verwendet werden. In diesem Feld wird der geeignete Dienst abgelegt.
  - **a-get-korrelation** = *auszuwertende Beziehung zwischen Attributen beim Abfragen der Attributwerte*
    Hier sind auch komplexere Berechnungsvorgänge erlaubt, die es ermöglichen sollen, ein Attribut, für das es keine direkte Entsprechung in einem anderen Modell gibt, aus mehreren anderen Attributen dieses Modells zu berechnen. Kann keine Korrelation für das betreffende Protokoll definiert werden, ist hier ein ϵ einzutragen.
  - **änderungsdienst** = *protokollabhängig*
    Wie beim Abfragen könnte es auch möglich sein, daß mehrere Dienste zur Änderung existieren, von denen der passende hier definiert wird.
  - **a-set-korrelation** = *auszuwertende Beziehung zwischen Attributen beim Setzen der Attributwerte*
    Auch beim Setzen kann eine komplexere Beziehung zwischen den Attributen bestehen. Existiert keine Korrelation, wird dies durch ein ϵ dargestellt.

- Für jede Operation $method_i$ wird eine

$$\begin{aligned} &\textit{Operationskorrelationsliste} \\ &= \{op_{\mathrm{korr}_1}, \ldots, op_{korr_k}\} \end{aligned}$$

definiert, mit den Tupeln op_{korr_i} = *(protokoll, o-korrelation)*:

  - **protokoll in** {*CMIP, SNMP, SNMPv2,...*}
    Hier wird für das MO, das die zu korrelierende Operation anbietet, das betreffende Protokoll abgelegt.
  - **o-korrelation** = *funktionaler Ausdruck*
    Analog zu den Attributen kann eine Operation bei einem MO auch durch eine Verkettung mehrerer Operationen auf einem anderen MO ausgedrückt werden. Ein ϵ zeigt an, daß keine Abbildung existiert.

- Für jede Meldung $report_i$ wird eine

$$\begin{aligned} &\textit{Meldungskorrelationsliste} \\ &= \{msg_{\mathrm{korr}_1}, \ldots, msg_{korr_k}\} \end{aligned}$$

definiert, mit den Tupeln msg_{korr_i} = *(protokoll, m-korrelation)*:

  - **protokoll in** {*CMIP, SNMP, SNMPv2,...*}
    Hier wird für das MO, von dem die zu korrelierende Meldung ausgeht, das betreffende Protokoll abgelegt.
  - **m-korrelation** = *funktionaler Ausdruck*
    Auch auf Meldungen kann es mehr als eine Reaktion geben. So kann z.B. eine eintreffende Meldung einer MOI erst dann als Meldung in der anderen Darstellung nach oben weitergereicht werden, wenn noch weitere Informationen aus dem Agenten abgefragt wurden. Mit ϵ wird eine nicht abbildbare Meldung angezeigt.

Beispiel 1 Diese Erweiterung würde bei dem obigen Beispiel für das Attribut *bBPPackage_bwbModulus* wie folgt aussehen:

Managed Object
Name *bandwithBalancingProcess*
Zugriffsauthentisierung ...
Attributliste
  attributname : *bBPPackage_bwbModulus*
    attributwert : *Integer*
    zugriffsart : *read-write*
    korrelationsliste
      protokoll : *SNMP*
      abfragedienst : *get*
      a-get-korrelation : *bmod.bmod*
      änderungsdienst : *set*
      a-set-korrelation : *bmod.bmod*
      ... (weitere Protokolle)
    ... (weitere Attribute)

Beispiel 2 Das SNMP–MO *bmod* ist nach der Erweiterung wie folgt definiert:

Managed Object
Name *bmod*
Zugriffsauthentisierung ...
Attributliste
  attributname : *bmod*

attributwert : *Integer*
zugriffsart : *read-write*
korrelationsliste
protokoll : *CMIP*
abfragedienst : *get*
a-get-korrelation :
bBP.bBPPackage_bwbModulus
änderungsdienst : *set*
a-set-korrelation :
bBP.bBPPackage_bwbModulus
... (weitere Protokolle)

Korrelation der Managed Object Instanzen

Bisher wurde noch nicht beachtet, daß MO's dynamisch instanziiert werden können. Das hat zur Folge, daß neben der statischen Zuordnung zwischen den Attributen von MO's auch eine dynamische Zuordnung der Instanzen stattfinden muß. Diese dynamische Zuordnung muß über die Namen der Instanzen erfolgen, da diese eindeutig sind. Dazu wird eine *Instanzenkorrelation* definiert als Instanz eines Meta–MO's. Das bedeutet, daß aus den Korrelationslisten für Attribute, Operationen und Meldungen jeweils eine Korrelation spezifiziert und mit konkreten Referenzen auf andere Meta–MOI's belegt wird.

Beispiel Die konkrete Instanz des Bandwith Balancing Modulus in SNMP (`bmod`) muß durch Anhängen von Index-Werten eindeutig identifiziert werden. Wird beispielsweise Bezug auf den Bandwith Balancing Modulus genommen, der für Bus 1 und Prioritätsstufe 0 gilt, muß `1.0` an das MO `bmod` angehängt werden. Diese Identifikation bezeichnet zusammen mit der Adresse des Agenten (`AgentID`) eindeutig die gewünschte Information. Unterstützt der Agent aber nur CMIP, so muß eine Instanzenkorrelation wie folgt gebildet werden:

Managed Object Instanz
Name *(AgentID, bmod.1.0)*
Zugriffsauthentisierung ...
Attributliste
attributname : *bmod*
attributwert : *Integer*
zugriffsart : *read-write*
protokoll : *CMIP*
abfragedienst : *get*
änderungsdienst : *set*
korrelation : *systemID = AgentID@ bandwithBalancingProcessID = 0 bandwithBalancingProcessPackage-_bwbModulus*

Wie man dabei sehen kann, müssen u.U. Instanzennamen erzeugt werden, wenn die Identifikation einer MOI bei der Umsetzung vorgenommen werden muß.

7.3.4 Integration der Protokolle

Nach der Beschreibung der Modellumsetzung stellt dieser Abschnitt die Dienstabbildung vor, die benötigt wird, um unabhängig vom jeweils verwendeten Protokoll eine Managementaufgabe zu lösen. Dazu wird nun ein *generisches Managementprotokoll* definiert, das alle Dienste umfaßt, die für die Implementierung einer Managementanwendung notwendig sind.

Ein generisches Managementprotokoll (GMP)

Weiter oben wurde das AM-API eingeführt, das vier Interaktionsmöglichkeiten zuließ. Diese bilden die Grundlage des GMP. Daraus lassen sich die in Tabelle 7.2 aufgeführten Dienste ableiten, die vom GMP erbracht werden müssen. Die Parameter sind dabei nur als Erklärung der Dienstprimitive anzusehen. Die konkrete Ausführung der Parameter wird in der Meta–MIB festgelegt. Außerdem wurde bei der Parameterliste von den Adressen abgesehen, die Manager und Agent identifizieren.

Der Einfachheit halber wird im folgenden davon ausgegangen, daß nur jeweils eine Managed Object Instanz (MOI) bzw. ein Attribut je Dienst betroffen ist.

Der *Operate_MO*–Dienst bezieht sich auf die Modifikation von MOI's als Ganzes. Als Operationen sind beispielsweise das Erzeugen oder Löschen einer Instanz möglich. Um sicherzustellen, daß die Modifikation erfolgreich abgelaufen ist, muß dieser Dienst bestätigt werden.

Mit dem *Set_Attribut*–Dienst kann ein Attribut einer MOI geändert werden. Dazu werden die MOI und das relevante Attribut identifiziert und der neue Attributwert übermittelt. Auch dieser Dienst muß bestätigt werden, um dem Manager den Erfolg der Änderung anzuzeigen.

Über den *Get_Attribut*–Dienst wird ein Attribut einer MOI abgefragt. Die Bestätigung dieses Dienstes enthält dann den gewünschten Attributwert, sofern die Anfrage zulässig war.

Der *Notify_Manager*–Dienst schließlich ist der einzige, der vom Agenten initiiert wird. Er wird verwendet, um dem Manager eine Ausnahmesituation anzuzeigen. Dazu gibt der Agent seine Identifikation an und benennt diejenige MOI, die diese Meldung ausgelöst hat. Ein weiterer Parameter spezifiziert den Grund für die Meldung, beispielsweise ein Zählerüberlauf oder eine unterbrochene Leitung. Dieser Dienst ist unbestätigt, da die Bestätigung quasi mit der Reaktion des Managers auf die angezeigte Ausnahmesituation erbracht wird.

Abbildung 7.1 zeigt das Zustandsübergangsdiagramm für die Dienstspezifikation. Dabei ist anzumerken, daß der Agent in dieser Abbildung ausschließlich symbolischen Charakter hat, da die Dienste des GMP immer in die eines konkreten Managementprotokolls umgesetzt werden, ehe sie an den Agenten gerichtet werden. Die Realisierung von GMP erfolgt also in einer Zwischenschicht zwischen der Managementanwendung und dem Manager.

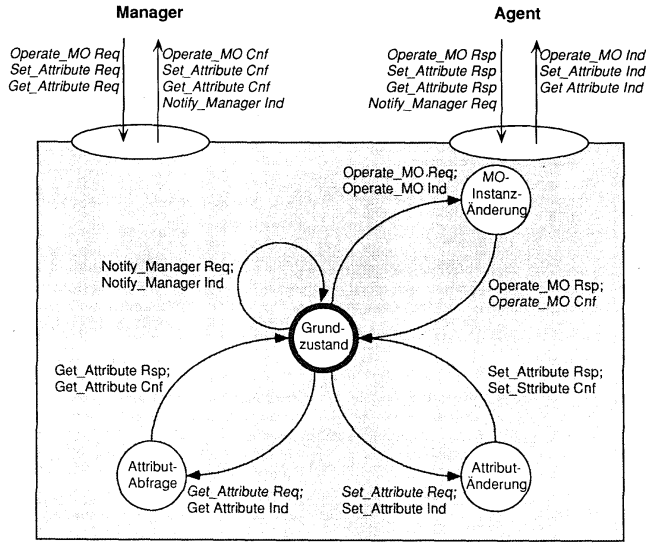


Abbildung 7.1: Zustandsübergangsdiagramm der Dienstspezifikation

Abschließend sei noch ein weiterer für das Management wichtiger Aspekt angesprochen: die Dienste gehen von einer eindeutigen *Manager* ↔ *Agent–Beziehung* aus. Um ein Management von beliebig großen heterogenen Netzen zu ermöglichen, ist es allerdings wichtig, daß eine hierarchische Managementstruktur eingeführt wird, d.h. ein Rechnerknoten kann sich auf der einen Seite wie ein Agent, auf der anderen Seite wie ein Manager verhalten. Da sich das GMP zwischen der Managementanwendung und dem konkreten Protokoll befindet, ist eine Unterstützung dieser hierarchischen Struktur ohne weiteres möglich. Dabei zu beachten ist nur, daß alle Dienste, die von einem als Manager agierenden Knoten ausgehen, durch das GMP übersetzt werden, während Meldungen, die der Knoten als Agent abschickt, erst beim adressierten Manager in das GMP umgesetzt werden.

Hin- und Rückabbildung standardisierter Managementprotokolle

Am Beispiel von SNMP soll nun gezeigt werden, wie eine solche Abbildung eines konkreten Protokolls auf das GMP aussieht. Dazu werden jeweils eine Hin- und Rückabbildung definiert, sowie zusätzliche Funktionen angegeben, die zur Dienstabbildung benötigt werden. Um die Darstellung nicht zu verkomplizieren, wird auf die Modellabbildung nicht mehr eingegangen, d.h. die Funktionen zur Realisierung der Abbildung zwischen Attributen, Methoden und Meldungen bleiben hier unberücksichtigt.

Abbildung 7.2 zeigt die Zustandsübergangsdiagramme für die Hin- und Rückabbildung zwischen SNMP und GMP. Tabelle 7.2 zeigt die jeweilige Dienstumsetzung.

Für die Hinabbildung des SNMP–*GetNext*–Dienstes ist eine Überführung nur durch eine zusätzliche Funktion zu erreichen. Diese Funktion, *LexSucc* genannt, liefert den lexikalischen Nachfolger der im *GetNext* angegebenen MOI, die bei SNMP durch den Agenten zurückgeliefert wird.

Ausgangsdienst	Zieldienst	Besonderheiten
SNMP-Dienst	GMP-Dienst	
Set	Set_Attribute	—
Get	Get_Attribute	—
GetNext	Get_Attribute	LexSucc
Trap	Notify_Manager	—
GMP-Dienst	SNMP-Dienst	
Set_Attribute	Set	—
Get_Attribute	Get/GetNext	Indetermin.
Modify_MO	Set	m. E.
Notify_Manager	Trap	—

Tabelle 7.3: Dienstumsetzung zwischen SNMP und GMP

Bei der Rückabbildung fallen zwei Besonderheiten auf:

Zum Einen ist der *Operate_MO*–Dienst von GMP in SNMP nicht verfügbar. Daher muß er — wenn überhaupt möglich — durch ein SNMP–*SetRequest* nachgebildet werden. Existiert diese Möglichkeit nicht, muß ein *MO.OperateRequest* sofort abgelehnt werden. Ob eine solche Nachbildungsmöglichkeit existiert und welcher Wert dann dem Aufruf mitgegeben werden muß, muß für jede MOI getrennt entschieden werden. Dazu dient die Information in der Meta–MIB. Zum Anderen kann ein Attributwert sowohl durch ein SNMP–*GetRequest* als auch durch ein SNMP–*GetNextRequest* abgefragt werden, was sich im Indeterminismus im Zustandsübergangsdiagramm widerspiegelt. Dieser Indeterminismus wird wieder durch den Eintrag in der Meta–MIB gelöst. Für jeden Attributwert wird dort für jedes Protokoll festgelegt, welches der passende Dienst zur Abfrage ist.

Plausibilitätsbetrachtungen

Die Definition von Hin- und Rückabbildung ist dann plausibel, wenn die Nacheinanderausführung der bei-

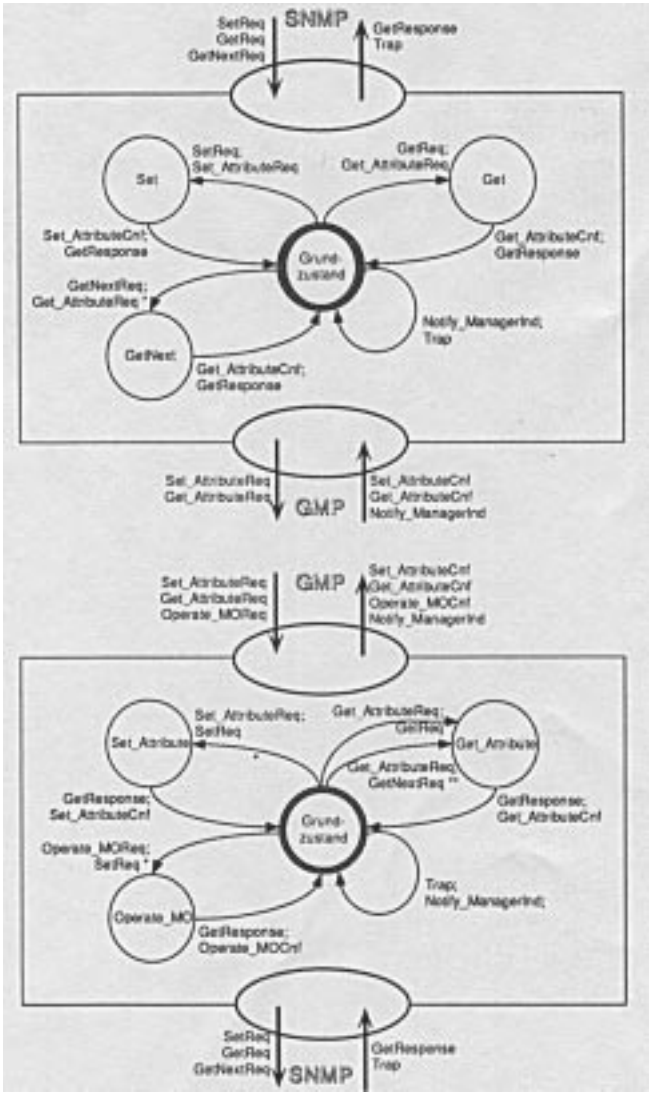


Abbildung 7.2: Erweiterte endliche Automaten der Dienstabbildung

Dienst	Parameter	Art	Erklärung
Operate_MO	Security_ Code, Object_Id, Operation, Parameter	bestätigt	Ausführung einer Operation, die von einem Managed Object bzw. dessen Instanz angeboten wird; Authentisierung des Managers; Identifikation der Managed Object Instanz; Art der Modifikation eventuell von der Operation benötigte Parameter
SetAttribute	Security_Code, Object_Id, Attribute_Id, Attribute_Value	bestätigt	Ändern eines Attributs einer Managed Object Instanz; Authentisierung des Managers; Identifikation der Managed Object Instanz; Identifikation des Attributs der Managed Object Instanz; Neuer Attributwert
GetAttribute	Security_Code, Object_Id, Attribute_Id	bestätigt	Abfrage eines Attributs einer Managed Object Instanz, die Rückgabe des Wertes erfolgt uber die Bestätigung; Authentisierung des Managers; Identifikation der Managed Object Instanz; Identifikation des Attributs der Managed Object Instanz
NotifyManager	Agent_Id, Object_Id, Reason	unbestätigt	Anzeige einer Ausnahmesituation, die bei einem Agenten aufgetreten ist; Identifikation des Agenten; Identifikation der Managed Object Instanz, welche die Anzeige angestoßen hat; genauere Beschreibung der Ausnahmesituation, z.B. Überschreitung eines Grenzwertes

Tabelle 7.2: Dienste eines generischen Managementprotokolls

den Abbildungen zur Identitätsabbildung führt. Anhand der Dienstabbildung zwischen SNMP und GMP soll der Plausibilitätsnachweis demonstriert werden.

Für jeden der Dienste Set, Get, GetNext und Trap muß nachgewiesen werden, daß er nach Anwenden der Hin- und Rückabbildung unverändert ist. Grundlage dafür ist die in 7.3 definierte Umsetzung der Dienste.

- Ein SNMP–*Set* wird auf ein GMP–*SetAttribute* abgebildet. Aus diesem wird in der Rückabbildung wieder ein SNMP–*Set*.
- Ein SNMP–*Get* bildet die Hinabbildung auf ein GMP–*GetAttribute* ab. Dieses kann in der Rückabbildung sowohl auf ein SNMP–*Get* als auch auf ein SNMP–*GetNext* abgebildet werden. Daraus wird durch Auswahl der ersten Alternative aber wieder ein SNMP–*Get*.
- Für das SNMP–*GetNext* kommt ebenfalls dieser Indeterminismus zum Tragen. Die Hinabbildung macht daraus ein GMP–*GetAttribute*. In diesem Fall führt die Auswahl der zweiten Alternative zur korrekten Rückabbildung.
- Ein SNMP–*Trap*, das durch die Hinabbildung auf ein GMP–*Notify_Manager* abgebildet wird, wird über die Rückabbildung wieder zum SNMP–*Trap*.

q.e.d.

Hiermit konnte die Plausibilität der Hin- und Rückabbildung recht einfach überprüft werden. Fehler in der Modellabbildung sind dagegen nicht so einfach nachzuweisen.

7.4 Sicherheitsaspekte und Fehlerbehandlung

7.4.1 Abbildung von Sicherungsmechanismen

Da der Datenverkehr vom Manager zum Agent zum Teil sicherheitskritische Aktionen hervorruft (das Ändern eines Attributwertes einer MOI in die falsche Richtung kann zum Zusammenbruch eines Teilnetzes führen), muß sichergestellt sein, daß jedes bei einem Agenten ankommende Paket auch wirklich so vom Manager abgeschickt wurde und immer noch seine Gültigkeit hat. Daher reicht es nicht aus, daß der Manager allein durch seine Adresse identifiziert wird, sondern er muß sich in jedem Paket authentisieren, und zwar so, daß es nicht möglich ist, anhand eines abgehörten Pakets ein gefälschtes zu erstellen, das vom Agenten nicht als solches erkannt wird.

Hierfür haben die Managementstandards eigene Mechanismen vorgesehen, die zum Teil nicht austauschbar sind. Daher muß auch zwischen den einzelnen Sicherungsmechanismen eine Abbildung existieren. Dazu wird eine Sicherungsabbildungsfunktion

$$\text{SAF}: \text{Manager-ID} \times \text{PS}_\text{M} \times \text{Standard-ID} \rightarrow \text{PS}_\text{A}$$

definiert, die als Eingabeparameter die Identifikation des Managers, die Parameter, die den Sicherungsvorkehrungen des vom Manager unterstützten Standards zu Grunde liegen, und eine Identifikation des vom Agenten unterstützten Standards erwartet. Als Ergebnis dieser Funktion werden die Parameter für die Sicherungsvorkehrungen bestimmt, die es dem Manager erlauben, auf den Agenten zuzugreifen.

7.4.2 Fehlerabbildung

Bislang wurde bei der Betrachtung der Modell- und Dienstabbildung von einer fehlerfreien Diensterfüllung ausgegangen. Dies kann nicht immer vorausgesetzt werden, da es zum einen zum Ausfall benötigter Komponenten kommen kann, und zum anderen eine Information beim Agenten nicht zugreifbar sein kann. Es lassen sich also zwei Fehlerarten unterscheiden:

1. Übertragungsfehler liegen dann vor, wenn der gewünschte Managementdienst durch das Protokoll nicht erfüllt werden kann, obwohl die mit dem Dienst spezifizierte Information beim Agenten vorliegt. Solche Fehler entstehen durch die Unzuverlässigkeit des verwendeten Übertragungsmechanismus. Beispiele sind ein verlorengegangenes *GetRequest* bei SNMP oder ein fehlgeschlagener Assoziationsaufbau nach einem CMIS–*M-Get*.
2. Als Modellfehler wird ein erfolgloser Zugriff auf ein MO beim Agenten bezeichnet. Dies kann dadurch geschehen, daß dieses MO beim Agenten nicht verfügbar ist, oder dadurch, daß die Managementoperation nicht zulässig ist, z.B. wenn auf ein nur lesbares MO schreibend zugegriffen werden soll.

Ähnlich wie bei der Dienst- oder Modellabbildung müssen etwaige Fehler, die in einer Architektur auftreten, in der anderen nachgebildet werden. Dazu werden zwei Abbildungsfunktionen benötigt. Dazu sei F_{MP} die Menge der möglichen Übertragungs- und Modellfehler des Managementprotokolls MP und I die Menge der globalen Fehlerindizes:

- Die **Fehlerverallgemeinerungsfunktion**

  $$\text{FVF}_\text{MP}: \text{F}_\text{MP} \rightarrow I$$

  weist jedem Fehler einen Fehlerindex zu. Diese Funktion muß für jedes Protokoll definiert werden. Es ist möglich, daß zwei konkrete Fehler auf einen gemeinsamen Index abgebildet werden, wodurch sich ein Informationsverlust ergibt. Dieser kann durch eine geeignete Wahl der Fehlerindexmenge I allerdings klein gehalten werden.

- Mit Hilfe der **Fehlerkonkretisierungsfunktion**

  $$\text{FKF}_\text{MP}: I \rightarrow \text{F}_\text{MP}$$

wird aus einem Fehlerindex eine dem Managementprotokoll entsprechende Fehlermeldung bestimmt. Das setzt voraus, daß die Abbildung für jeden möglichen Fehlerindex definiert sein muß. Dabei ist es aber erlaubt, daß mehrere Fehlerindizes auf einen konkreten Fehler abgebildet werden.

Tritt bei der Benutzung des Protokolls *MP1* ein Fehler auf, wird er mit Hilfe der Fehlerverallgemeinerungsfunktion auf einen Fehlerindex abgebildet. Aus diesem formt die Fehlerkonkretisierungsfunktion eine dem Managementprotokoll *MP2* entsprechende Fehlermeldung, die dann an die Anwendung weitergegeben wird.

7.5 Zusammenfassung

Um verschiedene Managementstandards integrieren zu können, ohne dabei die darauf aufbauenden Managementanwendungen zu beschränken, bieten sich verschiedene Ansatzpunkte an. Um die Heterogenität der Standards gegenüber den Anwendungen zu verbergen, ohne daß große Änderungen an den beteiligten Komponenten durchgeführt werden müssen und um flexibel hinsichtlich der Einführung neuer Managementstandards zu bleiben, bietet sich der Plattform–Ansatz an. Bei diesem wird zwischen den Managementanwendungen und den auf unterschiedlichen Standards basierenden Managern eine Konvergenzschicht eingeführt, die die Heterogenität verdeckt.

Zur Realisierung dieses Ansatzes wurden verschiedene Konzepte ausgearbeitet, die alle zur Integration unterschiedlicher Managementstandards zusammenarbeiten.

An der Schnittstelle zu den Anwendungen wird ein *allgemeines Management Application Programming Interface* (AM-API) definiert, welches als Grundmenge die fundamentalen Managementoperationen, also lesender und schreibender Zugriff auf die Managementinformationen, Auslösen einer Managementoperation und Entgegennahme von Ereignismeldungen, beinhaltet. Eine Erweiterung dieser Menge ist denkbar, wird aber für das Grundziel der Integration nicht gefordert.

Für die Verdeckung unterschiedlicher Monitortechniken sind die Agenten zuständig, wobei die Anbindung der Monitore an die Managed Objects nach der Anforderung an die Aktualität der Information zu geschehen hat. Bei einer Integration über eine Plattform zwischen Anwendung und Manager bleiben die Agenten jedoch unangetastet, so daß dieser Punkt nicht weiter untersucht wurde.

Für die unterschiedlichen Definitionen von Managed Objects muß hingegen eine Lösung gefunden werden. Diese wurde in Form einer Modellabbildung ausgearbeitet, in der für die Attribute, Operationen und Meldungen eines Managed Objects die jeweiligen Pendants in anderen Darstellungsformen festgelegt werden, was durch eine darstellungsunabhängige Definition der Managed Objects möglich ist. Zusätzlich müssen zwischen Managed Object Instanzen verschiedener Standards Korrelationen festgelegt werden.

Der Heterogenität im Bereich des Managementprotokolls wird durch eine Dienstabbildung entgegengewirkt. Zu diesem Zweck wurde ein *generisches Managementprotokoll* (GMP) definiert, das als Referenzprotokoll dient. In einer Diensthinabbildung werden für konkrete Managementprotokolle deren Dienste auf die des GMP abgebildet, die Dienstrückabbildung liefert die umgekehrte Richtung.

Abschließend wurden noch zwei für das Management wichtige Punkte untersucht:

Zur Erhaltung der Sicherheit, d.h. zur eindeutigen Autorisierung und Authentisierung des Managers sowie zur Verschlüsselung von Daten wurde eine *Sicherungsabbildungsfunktion* definiert.

Bei der Fehlerbehandlung mußten zwei Funktionen, eine *Fehlerverallgemeinerungsfunktion* und eine *Fehlerkonkretisierungsfunktion*, definiert werden, mit deren Hilfe eine Anwendung, die auf einem bestimmten Standard aufbaut, auch die auf einem anderen Standard basierenden Fehlermeldungen verstehen kann.

Mit diesen Konzepten kann ein umfassendes Management erreicht werden, wie es in Abschnit 7.2.1 klassifiziert wurde.

Zu diesem Ansatz wird in [Sei94] weiterhin ein Prototyp entwickelt, zu dem allerdings noch keine reale Implementierung existiert.

7.6 Literaturverzeichnis

[Sei94] [KP93]

Kapitel 8

Distributed System Management (DSM)

Jens Gaspar

8.1 Einführung

In der Industrie und im Forschungsbereich treten immer mehr verteilte Systeme auf. Durch großflächige Netzwerke können angeforderte Daten und Dienste sehr weit entfernt liegen. Man ist dabei nicht auf lokale Netze beschränkt, wie zum Beispiel interne Firmennetze. Große Datenmengen in anderen Netzen müssen auch zugänglich sein. Dadurch entstehen sehr große, komplexe verteilte Systeme.

Man kann dabei noch unterscheiden zwischen homogenen (gleiche Plattformen) und heterogenen Systemen, von denen wir im folgenden ausgehen.

Mit dem vermehrten Auftreten solcher Systeme und der damit verbundenen wachsenden Komplexität und erhöhten Abhängigkeit von Operationen, ist auch das Interesse am Management solcher Systeme gestiegen. Ein effektives Management ist gesucht, das unter anderem Fehlerfreiheit sichert, effiziente und sichere Operationen bereitstellt, es dem Anwender erlaubt gewünschte Berechnungen in möglichst kurzer Zeit auszuführen und die speziellen Fähigkeiten anderer Umgebungen vorteilhaft zu nutzen.

Der Begriff der *Manageability* (Verwaltbarkeit) rückt damit in den Vordergrund. Ein verteiltes System muß ausgerüstet sein mit Mechanismen, die ein effizientes Management von Betriebsmitteln gestattet. Heutige Systeme sind jedoch nicht dazu entworfen worden, verwaltbar zu sein im Hinblick auf die angesprochene Problematik. Verschiedene Anwendungen bzw. System-Konfigurationen haben unterschiedliche Anforderungen und brauchen andere Strategien zur Verwaltung. Manageability ist daher ein schwer faßbares Ziel.

8.2 Management durch Delegation

Verteilte Systeme arbeiten meist nach dem Client-Server-Prinzip. Prof. Yemini [Yem] hat Netzwerksysteme betrachtet, die mit verschiedenen Standards (SNMP/CMIP) arbeiten und ihre Schwächen und neue Ansätze zur Lösung herausgearbeitet.

Germán Goldszmidt hat zusammen mit Yemini daraus ein neues Paradigma für das Management in verteilten Systemen entwickelt.

8.2.1 Standardansätze

Anwendungen für Netzwerkmanagement sind meist organisiert als Client-Server Systeme. Die Programmierlogik von Anwendungen liegt beim Manager, die Daten, mit denen gearbeitet werden, liegen bei Agents. Agents sind also Server, die gewisse Dienste anbieten. Diese Dienste stellen einen Zugriff auf Gerätedaten (device data) und Grundkontrollfunktionen dar. Manager können Agents nur über einheitliche Schnittstellen aufrufen. Manager und Agents benutzen hierzu Managementprotokolle, um ihre Aktivitäten zu koordinieren.

Die Anwendungen sind logisch und physikalisch von den Daten und Geräten getrennt, die sie brauchen und kontrollieren müssen.

Der Client-Server Ansatz bietet also nur eine feste Funktionalität und fixe Schnittstellen. Die Dienste sind nicht veränderbar ohne eine Neu-Übersetzung, Neu-Installierung und Re-Instantiierung des Server-Prozesses. Durch die festen Schnittstellen sind Antworten nur auf vordefinierte Aufgaben erlaubt.

Es folgt die Auflistung der wichtigsten Kritikpunkte in Bezug auf SNMP/CMIP, die von Prof. Yemini [Yem] und Goldszmidt [GY92] beschrieben werden:

- *Warum versagen Netzwerke ?*
  Anfragen von Anwendungen werden durch das System durch mehrere Schichten hindurchgereicht. Dabei können Fehler auftreten. Treten nun sehr viele Anfragen zur gleichen Zeit auf, kann das zu einem Stau ausarten. Dieser führt dazu, daß einige Packets gelöscht oder zurückgeschickt werden, diese erneut verschickt werden und damit den Stau vergrößern. Durch weitere Verzögerungen verstärkt sich dieser Effekt noch. Solch ein Lawineneffekt kann ein Netzwerk zusammenbrechen lassen.

- Wie kann man ein *Versagen feststellen ?*

In einem verteilten System existiert eine gewaltige Anzahl an Prozessen, komplexe Interaktionen, Mehrfach-Schichten, etc. Es ist daher unklar, welche Komponenten überwacht werden sollen und wie man Rückschlüsse auf die Ursachen ziehen kann. Es ist auf jeden Fall erforderlich, Daten aus dem ganzen Netzwerk abzufragen, um festzustellen, ob es noch in Ordung ist (z.B.: Fehlerrate abfragen). Dazu ist aber ein häufiges Abfragen von Daten erforderlich. Ein zyklisches Abfragen der in Frage kommenden Daten *(polling)* würde das Netzwerk jedoch zusätzlich belasten. Außerdem ist durch das Polling ein zeitgleiches Vergleichen der Daten nicht möglich.

Solche Abfragen sollten flexibel verteilt werden können, nahe den zu überwachenden Größen. Mit den Standardansätzen ist dies jedoch nicht möglich.

- Ist *SNMP skalierbar ?*
  Die Zeitspanne, die für das Abfragen von Daten in einem großen, komplexen Hochgeschwindigkeitsnetzwerk gebraucht wird, überschreitet typischerweise die Kapazitäten der Netzwerkplattform.

  Wenn die Größe (Geräteanzahl), Komplexität (Anzahl der Variablen) oder die Geschwindigkeit des Netzwerks erhöht wird, oder wenn die Kommunikation eingeschränkt ist, wird das System schnell unbeherrschbar.

- *Semantische Heterogenität:* Wie bringt man Daten auf andere Plattformen ?
  Für den Transport von Daten von einer Plattform auf eine andere muß ein Protokoll vorhanden sein, das beide verstehen; und selbst wenn ein solches vorhanden ist, ist der Zugriff auf die Daten immer noch nicht möglich, denn das Protokoll vereinheitlicht die Syntax, nicht jedoch die Semantik von Datenzugriffen.

- *Micro-Management:*
  Bei nicht-trivialen Anwendungen, die zum Beispiel komplizierte Verknüpfungen von Daten ausführen, muß ein Manager einen Agent schrittweise durch die einzelnen Anweisungen geleiten, d.h. angeben, wie eine Aktion durchzuführen ist. Dies nennt man *Micro-Management*. Die dafür erforderliche Anzahl von Interaktionen zwischen Manager und Agent ist gewaltig, da feste Schnittstellen benutzt werden müssen. Verzögerte Antwortzeiten sind die Folge. Außerdem werden viele Rechenzyklen des Managers verbraucht. In kritischen Situationen führt dies zu ineffizienten und unzuverlässigen Systemen. Gerade bei Echtzeitanwendungen kann man sich das nicht leisten.

- *Zentralisierung* führt zu ineffektiver Verteilung von Funktionalität. Sie erzeugt fehlerträchtige Engpässe in der Kommunikation (Flaschenhals), und schränkt Echtzeitverhalten ein.

- Operationsdaten werden in *MIBs* gehalten:
  Das Datenformat einer MIB ist im voraus bestimmt und unabhängig vom Gebrauch. Der Nutzen von Daten ist schwer vorherzusagen, da er abhängig von zukünftigen Anforderungen bzw. Anwendungen ist.

  So kann es vorkommen, daß große Datenmengen angesammelt und gespeichert werden, ohne daß diese jemals benutzt werden.

- Client-Server Ansatz erfordert a-priori Wissen darüber, welche Operationen statisch in einer definierten *Schnittstelle* angelegt werden. Dadurch ist die Zusammenarbeit von Manager und Agent sehr starr ausgelegt.

8.2.2 Anforderungen an Management

Wie im vorigen Abschnitt aufgezeigt wurde, muß das Management für verteilte Systeme gewisse Anforderungen erfüllen:

- Viele verteilte Anwendungen fordern eine dynamische Änderung der Funktionalität ihrer Prozesse während sie ablaufen, d.h. die Funktionalität zwischen Client und Server muß dynamisch veränderbar sein. Damit ist eine Reaktion möglich auf sich dynamisch entwickelnde Anforderungen eines Systems oder einer Anwendung.

- Anstelle der Offenlegung von Informationen in MIBs (Management Information Base) und APIs (Application Interface) ist Modularität und Kapselung der Informationen anzustreben.

- Effizientes Management fordert ein Paradigma, das in einfacher Weise skalierbar ist, geeignet für Hochgeschwindigkeitsnetzwerke oder Netzwerke mit niedriger Bandbreite.

- Ein Informationstransfer auf andere Plattformen soll leicht vollziehbar sein.

- Um effektiv zu sein, müssen Manager das Ausmaß und die Komplexität von Informationen bewältigen, die große heterogene, verteilte Systeme charakterisieren. Dazu ist das Überwachen (Monitoring), die Interpretation und die Kontrolle von Betriebsmitteln (Hard- und Software) nötig.

- Organisationen brauchen Netzwerke, die sich selbst verwalten, um technisches Personal einsparen zu können.

- Geräteausfall, Leistungsineffizienz, unpassende Anforderung von Betriebsmitteln, Sicherheitsbelange und Accounting sollten entsprechende Behandlung finden.

- Sicherheit: Durch statistische Beobachtungen ist es leicht, kritische Netzwerkresourcen zu identifizieren und dort eine Störung zu verursachen und damit das Netzwerk lahmzulegen. Ein Mindestmaß an Sicherheit muß eingehalten werden:

  - Authentisierung des Zugriffs
  - Datenintegrität: Daten dürfen während des Transports nicht verändert werden
  - Datenzuverlässigkeit

8.2.3 Management durch Delegation mit Elastic Servern

Yemini und Goldszmidt [Gol, GY93a, GY92] haben ein Paradigma entwickelt, das die obigen Spezifikationen erfüllt: **Management durch Delegation mit Elastic Servern**

Elastic Server unterstützen ein Delegationsprotokoll, um die Funktionalität eines Servers dynamisch zu kontrollieren. Dafür werden einige Grundfunktionen bereitgestellt, mit denen man Programme beeinflussen kann. Elastic Server erlauben zusammengesetzte Anwendungen und sorgen für eine nahtlose Integration bereits vorhandener Tools.

Programme werden dabei als Daten betrachtet, die zwischen Prozessen verschickt und später instantiiert, d.h. ausgeführt werden können.

Management durch Delegation ermöglicht die Verteilung von Management Funktionalität, behält jedoch die Kontrolle über den Ablauf der delegierten Funktionen.

In den weiteren Abschnitten wird dies noch genauer erörtert.

8.2.4 Delegation von Programmen

Eine Anwendung, die auf einem Manager abläuft, kann Teilprogramme oder auch ganze Programme — in einer speziellen Skript-Sprache — an einen MAD Agent (MAD: Management durch Delegation) schicken. Diese Übertragung nennt man *Delegation*. Das übertragene Programm bezeichnet man als *delegiertes Programm (DP)*. Es werden also nicht mehr die benötigten Gerätedaten zu der Anwendung transportiert, sondern die Anwendung zu den Daten oder zumindest in deren Nähe. Das Programm wird dann in der Daten- bzw. Funktionsumgebung des MAD Agents — in der Laufzeitumgebung des Elastic Servers — ausgeführt. Dieser Zusammenhang ist in Abbildung 8.1 verdeutlicht.

Manager und Agents benutzen ein Delegationsprotokoll, um ein Programm zu übertragen und zu kontrollieren. Dieses Protokoll wird später näher beschrieben.

8.2.5 Elastizität

Elastizität ist die Fähigkeit, die Funktionalität eines Servers dynamisch zu erweitern oder zu reduzieren während er in Betrieb ist.

Elastizität ist nützlich für verteilte Anwendungen,

- die lange laufen, da während der Ausführung Änderungen durchgeführt werden können.
- die in heterogenen Umgebungen ablaufen, da eine "Übersetzung" anderer Protokolle leicht zu erreichen ist.
- die sich Veränderungen der Umgebung anpassen müssen, da dies ebenfalls dynamisch ausgeführt werden kann.
- die echtzeitfähig sein sollen, da durch eine dezentralisierte Managementlogik der Kommunikations- bzw. Interaktionsaufwand wesentlich geringer ist, und so durch geringere Antwortzeiten das Netzwerk besser ausgenutzt werden kann.

Elastic Server bieten einen einfachen, aber doch mächtigen Mechanismus, um verteilte Anwendungen zu erzeugen durch Verbinden bzw. Integrieren von unabhängig delegierten Programmen. Bereits vorhandene Tools können so einfach integriert werden.

Elastizität braucht ein Prozeß- und Kommunikationsmofell für erweiterbare Client-Server Interaktionen.

Die Elastizität wird durch eine Laufzeitumgebung ermöglicht, die aus einer Sammlung von Threads und Bibliotheksroutinen besteht.

8.2.6 Vorteile von Management durch Delegation

Management durch Delegation weist außer den aus der Spezifikation resultierenden Vorteilen, noch weitere auf, die nun besprochen werden.

- Es wird eine flexible Verteilung von Funktionen und dynamische Allokierung von Betriebsmitteln unterstützt.
- Es wird ein Grundgerüst geschaffen, das die Ausdruckskraft von Client-Server Interaktionen erhöht.
- Durch dynamische Allokierung von Funktionen in der Nähe, wo diese gebraucht werden, wird der Kommunikationsaufwand erheblich reduziert.
- In Netzwerken, in denen die Bandbreite für Übertragungen niedrig ist, kann die Autonomie von Geräten gesteigert werden.
  - Die Überlebenswahrscheinlichkeit verteilter Systeme kann erhöht werden.
  - Wenn das Netzwerk ausgebaut wird, oder mehr komplexe Geräte eingeführt werden, ist es relativ einfach eine Hierarchie aufzubauen, die eine regionale Autonomie gewährleistet.
  - Geräten ist es möglich, autonome Managementfähigkeiten zu erlangen, die abhängig sind vom Netzwerkstatus.
  - Geräte können den Umgebungsstatus dazu benutzen, geeignete Programme zu delegieren und zu instantiieren, um verschiedene Abstufungen von Autonomie zu etablieren.
- Delegation und Instantiierung können bequem geplant (scheduled) werden, so daß in Streßzeiten nur minimale Kommunikation notwendig ist.

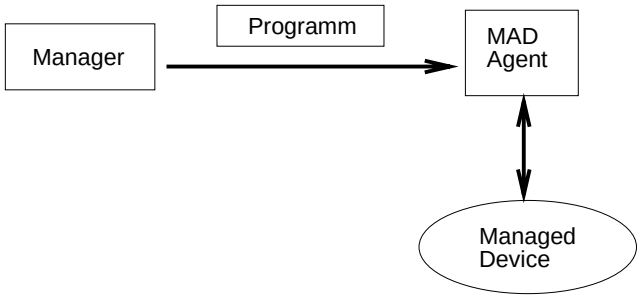


Abbildung 8.1: Delegation eines Programms

- Heterogenität:
  Programme können so entworfen werden, daß sie spezifische Operationsumgebungen und verschiedene Eigenheiten von bestimmten Betriebsmitteln handhaben können.

- Es ist kein einheitliches semantisches Modell für Gerätedaten nach Anpassung an verschiedene Plattformumgebungen notwendig. Delegation kann dazu dienen, Umwandlungen zwischen verschiedenen Managementprotokollen vorzunehmen.

- Managemententscheidungen können effizient durchgeführt werden, ohne daß ein Manager eingreifen muß.

- Kommunikation zwischen Clients und Management Programmen können durch existierende Protokolle unterstützt werden.

- Standards für Anwendungskommunikation können dazu vereinfacht werden, geeignete APIs (Application Interface) zu definieren, die minimale Schnittstellen umfassen.
  SNMP dagegen braucht spezielle MIBs, um sich auf veränderte Anwendungsmodelle einzustellen.

8.2.7 Anwendungen

Einige Anwendungen, für die man Management durch Delegation einsetzen kann, sind hier aufgelistet.

Ist ein Server überlastet oder versagt sein Host, kann eine Anwendung dynamisch einige ihrer Dienste an einen Elastic Server eines anderen Hosts delegieren.
Fehlende Rechenbetriebsmittel, etwa Prozessorzyklen, könnte die Funktionalität eines Servers herabsetzen. Durch Delegation einiger Funktionen an andere Server kann dies relativiert werden.

Software-Upgrades erfordern meist das Herunterfahren der entsprechenden Prozesse. Dann erst können neu kompilierte Versionen aufgespielt werden. Elastic Server gestatten dies während des Betriebs, d.h. ein (kurzfristiger) Ausfall des Servers ist nicht erforderlich.

Delegation kann dazu dienen, Umwandlungen zwischen verschiedenen Managementprotokollen zu generieren.
Die Heterogenität von Hardware und Software in verteilten Systemen erfordert Anwendungen, die mit mehreren Protokollen umgehen können. Durch Bereitstellen fester Unterstützung für alle möglichen Kombinationen von Protokollen, ist eine große Menge an zusätzlichem Code notwendig, ob er nun gebraucht wird oder nicht.
Elastic Server bilden einen Rahmen für Interoperabilität. Sogenannte Anwendungsbrücken können an bestehende Server delegiert werden, um zu erlauben, mit Clients zu operieren, die verschiedene Protokolle unterstützen. Der Codeaufwand wird dadurch erheblich vermindert.

Ist bei einem Fließband die Steuerlogik in einem Client zentriert, könnten Korrekturmaßnahmen zu spät ausgeführt werden, beispielsweise aufgrund unvorhergesehener Verzögerungen im Netzwerk. Durch Elastic Server können Clients die Instantiierung zeitkritischer Funktionen delegieren, womit auch die Anzahl an Interaktionen zur Kontrolle der Funktion reduziert wird.

8.3 Prozeß- und Kommunikationsmodell für Elastic Servers

Wie schon in Kapitel 8.2.3 und 8.2.4 erwähnt, werden Programme von einem Manager an einen MAD Agent delegiert. Dieser Abschnitt beschreibt diesen Vorgang detailierter.

Nach einer genaueren Betrachtung der delegierten Programme und des Protokolls für die Delegation, wenden wir uns dem Aufbau des Kernels eines Elastic Servers zu. Der Abschnitt schließt mit einem Anwendungsbeispiel (Komprimierung von Daten zur Entscheidungsfindung) ab, das in Kapitel 8.4 genauer beleuchtet wird.

8.3.1 Delegierte Programme

Bei der Delegation eines Programms wurde bisher stillschweigend angenommen, daß es dabei auch ausgeführt wird. Das ist aber nicht der Fall.
Wird ein Programm von einem Manager an einen MAD Agent delegiert, wird es in der Umgebung des Agents abgespeichert. Im Speicher des Agents liegt nun das

delegierte Programm. Um es auszuführen, muß der Manager es noch instantiieren. Es wird dann in der Laufzeitumgebung des Elastic Servers ausgeführt.

Es gibt verschiedene Möglichkeiten ein Programm zu starten (Typen delegierter Programme):

- Ein *delegierter Prozeß (complete process)* führt ein unabhängiges Programm aus mit einem Kontroll-Thread (Kontrollfaden) und einem eigenen Satz von Daten in der Umgebung des Elastic Servers. Es läuft als eigenständiger Prozeß.
- Eine *delegierte Prozedur (pure function)* kann aus dem Kontext eines Elastic Servers aufgerufen werden als lokale Prozedur oder als RPC (Remote Procedure Call). Die Prozedur kann einen Wert zurückgeben. Es treten jedoch dabei keine Seiteneffekte auf.
- Das "Programm" kann auch ein *delegiertes Programmobjekt* sein. Das ist eine Art passiver Server, der sich im Wartezustand befindet, und nur bei Erhalt einer Nachricht eine entsprechende Aktion ausführt.

Delegierte Programme werden in speziellen Management-Skript-Sprachen erstellt. Diese müssen die Fähigkeiten ausnutzen können, die der Softwareumgebung des Agents entsprechen. Dies kann zum Beispiel ein C-, C++- oder Pascal-Dialekt sein mit eingeschränktem Funktionsumfang:

- Zugangsbeschränkungen müssen berücksichtigt werden, zum Beispiel die Authentisierung für das Abfragen oder des Zurücksetzen eines Wertes.
- Heterogenität soll ausnutzbar sein

Ist die Hardwarearchitektur des Agents bekannt, an den das Programm delegiert wird, so kann auch der entsprechende Objektcode des Programms delegiert werden.

8.3.2 Delegationsprotokoll

Das Delegationsprotokoll stellt eine Schnittstelle mit Grundfunktionen zur Verfügung, die es Managern unter anderem erlaubt, Programme zu delegieren, zu instantieren, zu unterbrechen und damit fortzufahren, abzubrechen und zu entfernen, kurz, deren Ablauf zu kontrollieren. Das Protokoll wird von Managern, die delegieren und Elastic Servern benutzt.

Die Funktionen sind realisiert durch Threads in der Laufzeitumgebung der Elastic Server, die es ermöglichen, daß Clients und Agents (Elastic Server) bzw. Clients mit ihren Programminstanzen (DPI: Delegierte Programminstanzen) miteinander zu kommunizieren.

Die Schnittstelle umfaßt im einzelnen folgende Funktionen:

- `ES_Delegate(DP,&DPid);`
  Delegiere das Programm `DP` an einen Elastic Server. Das Programm wird dann im Speicher des Elastic Servers abgelegt und ein Handle auf das Programm wird in `DPid` (Identifikation für delegiertes Programm) zurückgegeben.
- `ES_Instantiate(DPid,Initparms,&DPIid);`
  Instantiiere das Programm, das durch `DPid` identifiziert wird, und bringe es dadurch zur Ausführung. Der zuständige Elastic Server liefert ein Handle auf die Instanz in Form einer ID zurück in `DPIid` Beim Aufruf können noch Parameter mitgegeben werden `Initparms`, zum Beispiel, ob die Instanz als eigenständiger Prozeß mit eigenem Adreßraum oder als Thread, angelegt werden soll, der im Adreßraum des Elastic Servers ausgeführt wird.
- `ES_Delete(DPid);`
  Das delegierte Programm mit der ID `DPid` wird aus dem Arbeitsspeicher des Elastic Servers gelöscht.
- `ES_Terminate(DPIid);`
  Die delegierte Programminstanz mit der ID `DPIid` wird abgebrochen.
- `ES_Suspend(DPIid);`
  Die delegierte Programminstanz mit der ID `DPIid` wird unterbrochen.
- `ES_Resume(DPIid);`
  Die Abarbeitung der delegierten Programminstanz mit der ID `DPIid` wird fortgesetzt.
- `ES_SendMsg(DPIid,Msg);`
  An die delegierte Programminstanz mit der ID `DPIid` wird eine Nachricht `Msg` geschickt.
- `ES_ReceiveMsg(DPIid,Msg);`
  Kommuniziere mit anderer Programminstanz `DPIid`.
- `ES_GetStatus(DPIid,&StatusRecord);`
  Frage Status der delegierten Programminstanz mit der ID `DPIid` ab und speichere das Ergebnis in `StatusRecord`.
- `ES_SetStatus(DPIid,StatusRecord);`
  Setze den Zustand der delegierten Programminstanz mit der ID `DPIid` auf den Wert von `StatusRecord`.

8.3.3 Kernel-Architektur

Die Kernelumgebung enthält Schnittstellen für Geräteoperationen, die den Zugriff auf lokale Daten und Funktionen erlauben. Diese Schnittstellen müssen nicht standardisiert sein.

Der Kernel unterstützt die Zugriffskontrolle, um die Betriebsmittel zu beschränken, auf die von verschieden delegierten Programminstanzen (DPIs) zugegriffen werden kann. Er implementiert Dienste, die eine Ausführungsumgebung für DPIs darstellen:

- Der *Scheduler* stellt Multitasking für DPIs zur Verfügung und eine Wakeup-Funktion, die einen wartenden Thread nach einem gewissen Zeitintervall aktiviert.

- *DPI Prozeß Manager*
- *IPC (Inter Process Communication)*
- *Naming Service*

Ein paar Begriffe, die schon zum Teil eingeführt wurden:

MAD: Management durch Delegation

MAD Agent: Ein Prozeß, der delegierte Programme empfängt.
Ein MAD Agent hat einen gewissen Grad an lokaler Autonomie. Er exportiert eine feste Anzahl von Diensten. Zusätzlich ist er aber ein Elastic Server, dazu in der Lage, seine Fähigkeit durch Delegation dynamisch auszubauen.

MAD Manager/Client: Ein Prozeß, der delegiert.
Ein MAD Manager kann selbst wieder ein MAD Agent für einen anderen Manager sein.

Die **Laufzeitumgebung eines Elastic Servers** wird nun genauer betrachtet. Abbildung 8.2 stellt die Zusammenhänge dar.

Nachdem ein Programm delegiert wurde, gibt der Agent ein Handle `DPid` an den Aufrufer zurück. Mit dieser ID wird das delegierte Programm im Agent angesprochen. Zu diesem Zeitpunkt liegt das Programm im Speicher des Agents. Dieses Handle kann mit anderen Prozessen geteilt (shared) werden, um mehrere Instanzen des gleichen Programms zu erzeugen. Mit der Instantiierung des Programms erhält der Client ein Handle `DPIid`, das die erzeugte Instanz identifiziert. Das Programm läuft nun asynchron in der Laufzeitumgebung des Agents ab, entweder als Thread, im Adreßraum des Agents, oder als eigenständiger Prozeß mit eigenem Adreßraum. Der Client, der das Programm delegiert hat erhält die Handles `DPid` und `DPIid` zur Kontrolle des Programms und der Instanz. Die Ausführung ist, wie eben erwähnt, unabhängig von der des Clients. Der Client hat jedoch die Kontrolle über den Ablauf des Programms.

Die Komponenten der Laufzeitumgebung und deren Zusammenhänge sind hier näher beschrieben:

- Der *Controller* initialisiert die Umgebung des Elastic Servers beim Start. Er instantiiert alle Threads der Komponenten, lädt und instantiiert vorher delegierte Programme.
  Während des normalen Betriebs ist der Controller für das lokale Management des Agents verantwortlich.
  Beim Terminieren, räumt er auf (Clean-Up).
- Das *Delegationsprotokoll* wird durch einen Thread implementiert. Es beschreibt die in Kapitel 8.3.2 aufgeführten Funktionen.
- Der *Speicher (Repository)* dient dazu delegierte Programme abzulegen (als Quellcode oder Objektcode), die dann instantiiert werden können.
- Der *Übersetzter für delegierte Programme (Translator)* ist dazu da, delegierte Programme zu überprüfen und zu übersetzen:
  1. Das delegierte Programm wird auf Legalität hin untersucht, d.h., ob es einer bestimmten Programmiersprache (Skript-Sprache) genügt.
  2. Liegt kein Fehler vor und enthält das Programm Quellcode, so wird es in Objektcode übersetzt.
  3. Der Objektcode des delegierten Programms wird im Speicher abgelegt. Es wird eine Datenstruktur angelegt für den Code und für zusätzliche Attribute, z.B. welche Sprache, welche Betriebsmittel nötig sind, etc.
- Der *DPI Manager* erlaubt es, den Zustand der delegierten Programme bzw. der Programminstanzen zu verändern: Instantiierung, Terminierung, Unterbrechung, usw.
- Der *Naming Service* stellt einen eigenen, protokollneutralen Weg dar, Namen an delegierte Programminstanzen zu binden.
- Ein Dienst für *IPC* (Inter Process Communication) unterstützt die asynchrone, lokale Kommunikation zwischen Thread-DPIs bzw. Prozeß-DPIs.
- Der *Scheduler* erlaubt präemptives Planen (Schedulen) für Thread-DPIs.

DPIs und OCPs (Observation and Control Points, siehe unten) werden als unabhängige Prozesse in der Laufzeitumgebung des Agents ausgefhrt.

Um dynamisch verteilte Anwendungen zu erstellen, sind **Verbindungen** zwischen Clients, Servern und delegierten Prozessen erforderlich. Die Instantiierung und die **Zugriffskontrolle** auf andere Prozesse und Daten, sowie die interne Struktur eines MAD Agents und die Beziehungen zwischen ihren Komponenten ist in Abbildung 8.1 dargestellt.

Instantiierte Prozesse oder Programme (DPIs) können Verbindungen aufbauen, um Nachrichten zu senden und zu empfangen, oder um Funktionen im Agent selbst oder in anderen Servern aufzurufen.
Kommunikation zwischen Thread-DPIs ist direkt möglich über Shared-Memory. Die Kommunikation zwischen Prozeß-DPIs ist implementiert über BSD Socket-Schnittstellen.

Wie in Abbildung 8.3 zu sehen ist, bekommen die Instanzen (DPIs) ihre Daten von entfernten verwalteten Objekten über sogenannte *OCPs (Observation Control Points)*. Ein OCP ist ein Prozeß in einer MAD Agent-Umgebung, der einen Satz von Diensten anbietet, um es DPIs zu ermöglichen mit Managed Objects zu kommunizieren. Eine OCP-Schnittstelle versteckt die Details des Managementprotokolls; es kann außerdem ein gegebenes Protokoll reproduzieren. So ist ein einfacher Zugriff auf andere Plattformen möglich: Die relevanten

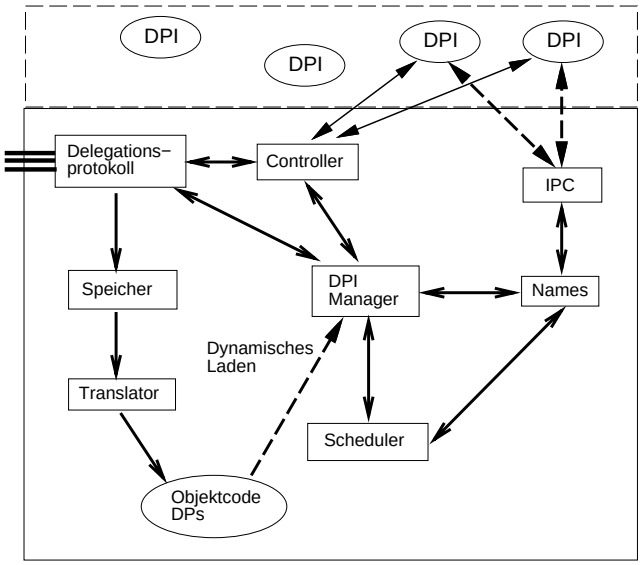


Abbildung 8.2: Laufzeitumgebung eines Elastic Servers

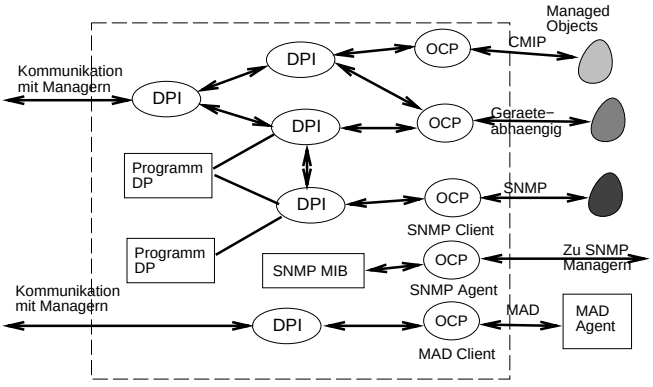


Abbildung 8.3: Delegierte Programminstanzen

Informationen sind also gekapselt (Information hiding). Diese Dienste umfassen:

- Repräsentation von Managed Objects, von Ereignissen oder von Signalen und macht diese anderen DPIs zugänglich.
- Der Status von Managed Objects kann abgefragt werden.
- OCPs können Werte von Attributen lokal speichern oder häufig benutzte Informationen in einem Cache ablegen.

8.3.4 Anwendungsbeispiel: Komprimierung von Informationen

Für Managemententscheidungen müssen aus den vorhanden Daten, die in Managed Objects (MIBs) abgelegt sind, bestimmte Informationen extrahiert und verknüpft werden. Diese Auswahl und Verknüpfung der Daten nennt man *Komprimierung*[1]. Die Entwicklung effektiver Technologien für Komprimierung in Echtzeit stellt ein zentrales Problem von Netzwerkmanagement dar.

Management durch Delegation sieht dafür den folgenden Ansatz vor:
Die Komprimierung wird durch eine *Health-Funktion* erledigt, auf die in Kapitel 8.4 noch detailierter eingegangen wird. Diese Funktion wird meist als Linearkombination von MIB-Daten zu einem Index-Wert umgerechnet, der den Netzwerkstatus beschreibt. Dieser Index wird für Managemententscheidungen verwendet. Dabei ergeben sich die folgenden Probleme:

- Health-Funktionen können nicht Teil einer statischen MIB sein, da sie abhängig sind von Konfiguration, Installation, Site (Standort) und Zeit.
- Health-Funktion ist ungeeignet für zentralisierte Managementplattformen nach dem traditionelles Client-Server-Prinzip, da dies in exzessives Polling (zyklisches Abfragen von Daten) ausarten würde.
  Beispiel: Geräte werden alle 0,1 Sekunden abgefragt und die Anzahl der untersuchten Schnittstellen sei 200. Dann wären 2000 SNMP Anfragen pro Sekunde notwendig.
- Das Ziel ist daher, Daten direkt an der Quelle zu komprimieren, damit Pollen vermieden wird; denn dadurch wird die Berechnung der Health-Funktion beeinflußt, da die Daten Werte nicht zeitsynchron abgefragt werden. Verhindert wird Polling durch Delegation von Health-Funktionen an MAD Agents.
- Ein direkter oder zumindest räumlich naher Zugriff auf Operationswerte und minimale Antwortzeiten erlauben auch eine hohe Genauigkeit.
- Durch Kombinieren von Health-Indikatoren können große Mengen an Daten effektiv komprimiert werden.
- Agents, die Health-Funktionen berechnen, können sofort Gegenmaßnahmen einleiten, ohne daß ein Eingreifen eines Managers erforderlich wird.
  Beispiel: Wenn eine Leitung zu oft benutzt wird, kann ein Überwachungsprozeß die Anfragen durch andere Leitungen weitergeben, oder auch die Leitung (temporär) vom System abkoppeln.

Goldszmidt [GY93b] hat eine Anwendung entwickelt, die einen *Health-Index* eines verteilten Systems berechnet. Es werden dabei dynamisch Überwacher (OCPs) erzeugt, die das Verhalten von Netzwerkelementen ermitteln und in einem Wert umrechnen mit Hilfe von verschiedenen Health-Funktionen. Die errechneten Index-Werte können in übergeordneten Health-Funktionen wiederum kombiniert werden zu einem globalen Index-Wert, dem *Health-Index*. Dieser stellt eine quantitative Repräsentation des Systemzustandes dar, der beispielsweise vom System graphisch angezeigt werden kann.
Die Anwendung bedient sich dabei der Vorteile der Elastic Server Fähigkeit, während der Ausführung dynamisch die Funktionalität der Prozesse zu erweitern oder zu reduzieren.
Für ein gegebenes System muß die Bewertung der Health-Parameter natürlich dynamisch varriert werden, da sich die Umgebung über die Zeit hinweg ändert.
Beispiel: Die Anzahl der Benutzer an Terminals.

8.4 Health-Anwendung

Managemententscheidungen, wie z.B. Abkoppeln eines Gerätes, werden abhängig vom Systemzustand getroffen. Ein geeignetes Maß zu finden bzw. eine Auswertungsfunktion zu entwickeln, die den "Gesundheits"-Zustand eines heterogenen, verteilten Systems beschreibt, ist eine schwierige Aufgabe. Eine solche Funktion berechnet durch Verknüpfen bestimmter Daten (Komprimierung von Daten, siehe Kapitel 8.3.4) einen oder mehrere Index-Werte, die den Zustand des Systems wiedergeben. Solch ein Index ist vergleichbar mit den Indizes aus der Wirtschaft: DAX (Deutscher Aktien Index), Dow Jones Index oder Inflationsrate.

Je nach dem Wert des Index werden bestimmte Managementaktionen ausgeführt. Einen solchen Index-Wert nennt man *Health-Index*, die entsprechende Funktion dazu heißt *Health-Funktion*.

Wann ein verteiltes System "gesund" ist, ist jedoch von System zu System verschieden, da dies von der jeweiligen Installation abhängig ist. Man denke dabei zum Beispiel an Konfiguration von Geräten, administrative Absicherungen oder an den Nutzungsgrad bzw. Verzögerungszeiten.

Was für eine akademische Abteilung ausreicht, mag inakzeptabel sein für die Intensivstation eines Krankenhauses.

[1] ... nicht im Sinne von Packen.

Die Parameter für die Health-Funktion sind also system- und zeitabhängig.

Health-Funktionen können deshalb auch nicht statisch definiert werden, sondern sollten dynamisch an Agents gebunden werden können.

Die Berechnung der Health-Indizes basiert auf Informationen, die man von *Überwachern*, den *OCPs (Observation and Control Points)*, erhält, wie in Abbildung 8.4 zu sehen ist. Die OCPs beobachten das Verhalten von Managed Objects und erzeugen Diagnosewerte. Die Überwacher (OCPs) müssen spezifische Details des Netzwerkes kennen, z.B. wo ein Fehlerzähler zu finden ist. Die Health-Anwendungen werden so geschützt vor geräte- und protokollspezifischen Details.

Wie auch in Abbildung 8.4 zu sehen ist, erhalten Health-Objekte Berichte (Reports) von den OCPs, die Informationen enthalten, woraus dann ein Health-Index berechnet wird. Schließlich wird ein endgültiger Index-Wert an den Manager übergeben.

Die von Goldszmidt [GY93b] beschriebene Health-Funktion wird dazu benutzt, die Leistung eines verteilten Systems zu berechnen. Außerdem werden durch eine Bewertung Fehler aufgespürt und korrigiert. Die dynamische Zusammensetzung und Konfigurierung der Health-Anwendung wird erreicht durch Benutzung von Elastic Servern. Komponenten der Health-Anwendung werden dabei dynamisch delegiert und bieten Dienste an, die von Managern aufgerufen werden können, um Konfigurationsänderungen zu unterstützen.

So ist etwa eine Liste von Prozessen, mit denen eine delegierte Programminstanz kommunizieren darf, dynamisch konfigurierbar.

8.4.1 Health-Funktion

Eine Health-Funktion berechnet einen Health-Index aus ihren Eingabewerten. Diese können wiederum Health-Indizes sein aus untergeordneten Health-Funktionen.

Ein Index berechnet sich oft durch eine Linearkombination von Variablen (z.B. Zähler, Fehlerraten), die über OCPs geliefert werden (siehe Kapitel 8.4). Jede Variable trägt so ihren Teil zum Health-Index bzw. Systemzustand bei.

Beispiel: Gegeben seien zwei MIB-Zähler:

`ifInOctets` = Anzahl der *erhaltenen* Bytes durch eine Schnittstelle seit Geräte-Initialisierung

`ifOutOctets` = Anzahl der *gesendeten* Bytes durch eine Schnittstelle seit Geräte-Initialisierung

Die Nutzung der Schnittstelle zum Zeitpunkt t kann definiert werden als:

$$U(t) = \frac{(\texttt{ifInOctets} + \texttt{ifOutOctets}) * 8}{\texttt{ifSpeed} * t * 100}$$

Das Maß $U(t)$ gibt den Nutzungsdurchschnitt über ein Zeitfenster seit Initialisierung an.

Üblicherweise liefert nur die Differenz im Verhältnis zum aktuellen Wert einen brauchbaren Indikator für den Netzwerkzustand.

Ein nützlicher Indikator des unmittelbaren Netzwerkzustands ist durch die Ableitung $u(t) = U'(t)$ gegeben ($U(t)$ aus obigem Beispiel). Diese kann durch häufiges Abfragen der entsprechenden Variablen und Differenzenbildung von $U(t)$ approximiert werden.

Beispiel: Unmittelbare Fehlerrate, Prozentsatz von Eingabefehlern zu gelieferten Packets:

$$E(t) = \frac{\texttt{ifInErrors}}{(\texttt{ifInUcastPackets} + \texttt{ifInNUcastPkts})}$$

Auch hier ist nur $e(t) = E'(t)$ von Interesse.

Wie schon erwähnt können einzelne Indikatoren wiederum linear kombiniert werden zu einer neuen Health-Funktion.

Beispiel: $U(t)$, $E(t)$, sowie $u(t)$ und $e(t)$ aus obigen zwei Beispielen. Dann kann man eine neue Health-Funktion wie folgt definieren:

$$\mathcal{H}(\vec{e}, \vec{u}) = \vec{A}\vec{e} + \vec{B}\vec{u}$$

mit $\vec{e}$ aus $e(t)$ für alle Indikatoren bzw. analog für $\vec{u}$ und mit den Gewichtsvektoren $\vec{A}$ und $\vec{B}$.

8.4.2 Schwellwertentscheidungen

Netzwerkmanagemententscheidungen können beschrieben werden durch *Schwellwertentscheidungen* — Bool'sche Funktionen über der Menge der Management Überwachungen. Anwendungen benutzen Entscheidungsprozesse, um Aktionen auszulösen.

Zum Beispiel, wenn die Fehlerrate für das Lesen von Daten über eine Leitung einen gewissen Wert übersteigt, werden Maßnahmen ergriffen.

Oft muß jedoch eine ganze Anzahl an Indikatoren gleichzeitig betrachtet werden.

Eine hohe Fehlerrate kann auch von erhöhtem Netzverkehr stammen. Dies muß entsprechend Berücksichtigung finden.

Eine linear gewichtete Größe ist eine effiziente Methode, um Variablen in einer Schwellwertfunktion zu verwenden: Sei $\vec{x} = (x_1, \ldots, x_k)$ eine Sammlung von Variablen, die zur Entscheidung beitragen sollen und $\vec{w} = (w_1, \ldots, w_k)$ ein Satz von Gewichten. Die Health-Funktion h ist dann definiert durch das Skalarprodukt:

$$h(\vec{x}) = \vec{w}\vec{x} = \sum_{i=1}^{k} w_i x_i$$

Eine *lineare Schwellwertentscheidung* ist demnach:

$$\mathcal{H} = \{\vec{x} \mid h(\vec{x}) \geq 0\}$$

Werte von $\vec{x}$ in $\mathcal{H}$ definieren Ausnahmeereignisse, die eventuell Probleme andeuten.

Ein Vergleich auf einen anderen Wert, etwa $\geq c$ ist einfach möglich durch Erweiterung der Vektoren $\vec{w}$ und $\vec{x}$ auf $\vec{w} = (w_1, \ldots, w_k, -1)$ bzw. $\vec{x} = (x_1, \ldots, x_k, -c)$.

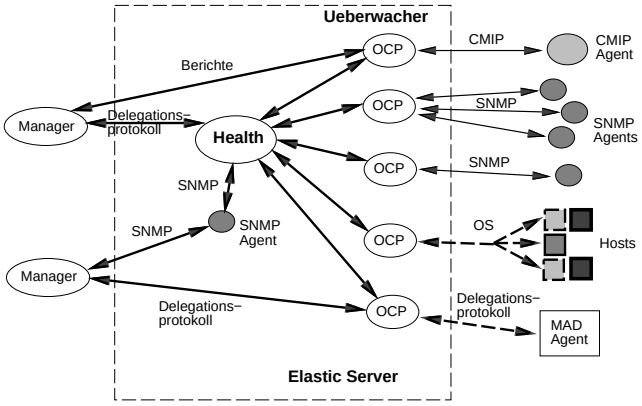


Abbildung 8.4: Health-Anwendung

Die Summe wird dann natürlich über $k+1$ Elemente gebildet.

Diese Methode ist einfach und schnell auszuwerten. Sie ist außerdem attraktiv, da $\mathcal{H}$ als Eingabe für weitere Health-Funktionen dienen kann, da $\mathcal{H}$ die Fehler-Indikatoren enthält.

Managementaktionen müssen nur manchmal durchgeführt werden, wenn zum Beispiel unpassendes Verhalten ber eine gewisse Zeit bestehen bleibt oder wenn es periodisch auftritt. Das mu durch geeignete Operatoren in die Health-Funktion eingehen.

Indikatoren könnnen variieren abhängig vom Netzwerk und der Zeit. Die Health-Funktion muß daher angepaßt werden durch Veränderung der Gewichte. Dies kann geschehen durch Lösungsansätze, die vom Perzeptron-Training bekannt sind, etwa Algorithmus der kleinsten Quadrate, denn nichts anderes liegt hier vor als ein einschichtiges Perzeptron. Durch das Training werden schließlich die Gewichtswerte optimal angepaßt.

Sollen mehrere Health-Funktionen bercksichtigt werden, hat man ein mehrschichtiges Perzeptron, das auf entsprechende Weise trainiert werden kann. Trainingsalgorithmen findet man in der einschlägigen Literatur.

8.5 Kritikpunkte

Nachdem die wesentlichen Merkmale des "Management durch Delegation"-Ansatzes besprochen wurden, sollen noch einige Kritikpunkte aufgezeigt werden.

- **Zugriffsrechte:**
  Eine Identifizierung ist notwendig zur Einhaltung der Sicherheitsbestimmungen.

  Unterschiedliche Abstufungen des Zugriffs müssen realisiert werden, ob zum Beispiel ein Objekt nur ein Program ausführen darf, oder auch Informationen abfragen darf, oder ob ein Objekt völlige Kontrolle über den Agent hat.

- Zugriff auf Informationen anderer Managed Objects dürfen nur über OCPs abgewickelt werden. Ein willkürlicher Zugriff wäre fatal $\Rightarrow$ **Zugriffskonflikte**

  OCPs müssen in der Umgebung des MAD Agents allokiert und gewartet werden.

- **Indeterminismus von Managementaktionen:** Betrachte beispielsweise einen Agent, an den einige Programme delegiert wurden, die alle auf ein Ereignis E warten. Tritt E dann ein, welche Aktion wird dann zuerst ausgeführt? ... Die Reihenfolge kann das Resultat ganz entscheidend beeinflussen.

  Beispiel: Ein X.25 Kontroller enthält einen Agent, der zuständig ist für Kontrolleroperationen. An den Agent wurden 3 Skripts von verschiedenen Managern delegiert, die ausgeführt werden sollen, wenn das Ereignis "Puffer voll" eintritt. Die Skripten sind:

  1. Starte Diagnoseprogramm, um möglichen Link- oder Kontrollerfehler aufzudecken und starte Kontroller neu
  2. Werte Leistungsparameter aus und begrenze lokale Benutzung von Betriebsmittel
  3. Gib Inhalt von Puffer aus und führe Reset aus

  Indeterminismen sind vermeidbar durch die Bedingung, daß Programme zum gleichen Ergebnis führen, unabhängig von der Ausführungsreihenfolge. Andernfalls muß die Reihenfolge durch Prioritäten festgelegt werden.

  Management-Aktionen können sich gegenseitig beeinflussen. Das stellt ein Risiko für Fehler dar. Fehler treten meist dann auf, wenn Auswertung eines Ereignisses oder einer Aktion Änderungen in anderen Managed Objects auslöst, die ebenfalls benutzt werden. Solche Fehler können sich fortpflanzen und schlimmstenfalls alles blockieren.

8.6 Zusammenfassung

Management durch Delegation: Dieses Forschungsgebiet steht noch ziemlich am Anfang. Es zeigt jedoch auf, daß bisherige Standards keine "Alleskünstler" sind.

Der Weg zu sich selbst verwaltenden Netzwerke ist noch weit.

Ein Prototyp mit einer MAD Umgebung ist implementiert. Er stellt eine Umgebung zur Verfügung, um delegierte Anwendungen zu entwickeln, zu verwalten und zu debuggen durch Delegation. MAD Agents schließen dabei Netzwerkmanagement mit ein.

8.6.1 Prototyp

Aktueller Status des Prototyps; Implementiert sind:

- Prozeß Manager für delegierte Programminstanzen
- Delegationsprotokoll
- Speicher zum Ablegen/Laden von delegierten Programmen
- Übersetzer für delegierte Programme, geschrieben in einem C- oder C++-Dialekt
- SNMP basierter OCP, der es einem MAD Agent erlaubt, ebenso als SNMP Agent zu arbeiten
- IPC Dienst zum Versenden von Nachrichten
- einfacher MAD Agent Kontroller
- zwei Benutzerschnittstellen: eine unter X und ein einfaches Text-Terminal

Außerdem wurden einige simple Anwendungen geschrieben, die auf MAD Agents als delegierte Programme laufen. Der Prototyp kann Management Skripts empfangen, übersetzen, linken und ausführen (asynchron).

8.6.2 Kosten

Ein großer Vorteil von Management durch Delegation ist der niedrige Kommunikationsaufwand (von Nachrichten). Die Flexibilität und die gesteigerte Ausdruckskraft ist jedoch nicht umsonst. Es sind mehr Rechenzyklen und mehr Speicher in MAD Agents erforderlich.

Die Kosten dafür kann man grob in zwei Kategorien einteilen:

- feste Komponenten (eingebauter MAD-Support)
- variable Teile (OCPs, DPs, DPIs)

Da durch schlechtes Netzwerkmanagement erhöhte Kosten entstehen, die Kosten für Rechengeräte jedoch stetig abnehmen, insbesondere von Mikroprozessoren, kann man diese Einbußen rechtfertigen.

8.7 Literaturverzeichnis

[Yem] [Gol93] [GY93b] [Gol] [GY93a] [GY92]

Kapitel 9

Distributed Computing Environment

Li Da Ren

9.1 Einleitung

Das Distributed Computing Einvironment (DCE) der Open Software Foundation (OSF) stellt eine Reihe von Softwarekomponenten bereit, um die Erstellung von verteilten Anwendungsprogrammen auf heterogenen Rechnerknoten zu unterstützen. Dieser kleine Aufsatz beschränkt sich hauptsächlich auf die Themen, Gesamtsystemarchitektur des DCE sowie Teilarchitektur seines Komponenten.

9.1.1 Begriffserklärungen

Im folgenden werden zwei wichtigen Begriffe, die sich um DCE beziehen, nämlich verteilte Anwendung sowie verteiltes System erklärt. Dadurch kann die Verständigung dieses Aufsatzes erleichert zu werden.

Verteilte Anwendung

In den letzten Jahren gewinnen verteilte Anwendungen in den umfangreichen Gebieten immer zunehmend an Bedeutung. Sie lassen sich beschreiben als Programme einer Menge von Modulen, die auf unterschiedlichen Rechnern platziert sind und während ihrer Verarbeitung über ein Kommunikationsnetz interagieren. Bild 9.1 zeigt ein Beispiel einer verteilten Anwendung im Bereich der Büro- und Fertigungs- Automatisierung. In diesem verteilten System können z.B verteilte Anwendungen wie Projektplanung, Buchhaltung, Fertigung und etc. unterstützt werden.

Verteiltes System

Ein verteiltes System setzt sich aus einer Menge der beteiligten Rechnerknoten, dem Kommunikationsnetz sowie der systemnahen Software zur Realisierung der verteilten Anwendungen zusammen. Ebenfalls spiegelt sich diese Definition intuitiv im Bild 9.1. Heutzutage werden außer standardisiertes ISO/OSI Referenzmodell einige spezielle verteilte Systeme entwicklet, wozu SUN ONC, OMG/CORBA, ODP-Referenzmodel und DCE gehören. Im folgenden werden wir uns nur mit DCE weiter beschäftigen.

9.1.2 Historische Entwicklung des DCE

Die Open Software Foudation (OSF) mit Hauptsitz in Cambridge, Massachussetts, wurde im Jahre 1988 gegründet mit der Zielstrebung, herstellerneutrale Software-umgebungen zu schaffen, die globale Interoperabilität verschiedener Systeme ermöglichen. Die Organisation OSF umfaßt als Mitglieder Rechnerhersteller, Softwarehäuser, industrielle Anwender, öffentliche Institute, Forschungsinstitute und Universitäten. Im Jahre 1989 konnten Technologievorschläge für das verteilte System DCE eingereicht werden. Erst im Mai 1990 wurde die Auswahl der verschiedenen Vorschläge abgeschlossen. Danach wurde die Integration der Technologiekomponenten durchgeführt. Anfang 1992 stand die erste Vision des DCE zur Verfügung.

9.1.3 Aufgaben der Dienstelementen von DCE in den Anwendungen

Das DCE stellt ein spezielles verteiltes System dar, das relativ umfangreiche Software-Mechanismen zur Unterstützung der verteilten Anwendungen bietet. Es setzt sich aus einer Reihe von Systemkomponenten mit zugehörigen Werkzeugen und Laufzeitbibliotheken zusammen und stellt Softwarelösungen für die in den verteilten Anwendungen entstehenden Probleme.

Kommunikation und Heterogenität

In DCE Umgebung dient Remote Procedure Call (RPC) zur Kommunikation zwischen Modulen. Auf Basis integrierter Datenkonvertierungsmechanismen wird übrigens die Kommunikation zwischen heterogenen Systemen unterstützt.

Modularisierung und Systemgröße

Mit wohldefinierten Schnittstellen und Typprüfung bei RPC-Kommunikation wird eine strikte Modularisierung der Anwendungssoftware erzwungen. Dadurch kann auch große verteilte Anwendung leichter gehandhabt werden.

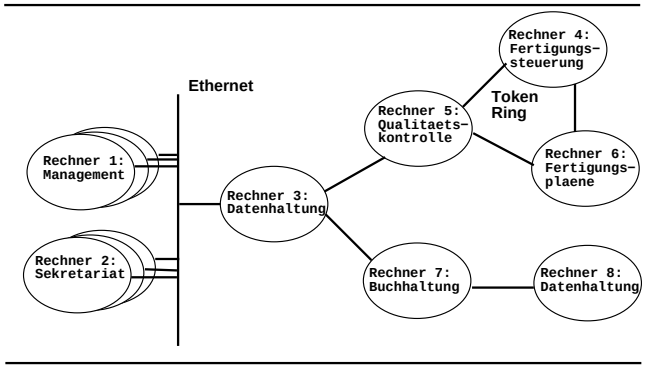


Abbildung 9.1: Beispiel einer verteilten Anwendung

Parallelität und Synchronisation

DCE bietet Thread Service und Distributed Time Service zur Parallelisierung sowie Synchronisation in den verteilten Anwendungen.

Sicherheit

Die Probleme des Zugriffs- und Datenschutzes werden in DCE durch einen Authentisierung- und Autorisierungsdienst (Security service) gelöst.

Verteilte Namensverwaltung

Die verteilte Namensverwaltung wird durch miteinander integrierte Dienste erbracht, den Cell Directory Service (CDS) und den Global Directory Service.

Verteilte Datenhaltung und PC-Integration

Das DCE verfügt über Distributed File System (DFS) und Diskless Support Service (DSS) zur verteilten Datenhaltung. Besonders unterstützt DSS die Dateiverwaltung im Rechner ohne Hintergrundspeicher. Schließlich können Personal Computer mittels des DCE PC Integration (PCI) in eine DCE Umgebung einbezogen werden.

9.2 Architektur des OSF DCE

Im diesen Abschnitt wird zuerst die Gesamtarchitektur des DCE und die Struktur der Teilkomponenten dargestellt. Danach wird das Zusammenwirken der Teilkomponenten beleuchtet.

9.2.1 Schichtenarchitektur

Die Architekturmodell des DCE ist im Bild 9.2 skizziert. Hier werden die Beziehungen zwischen verschiedenen Komponenten deutlich gezeigt. Als Voraussetzung für verteilte Anwendung und DCE-Komponenten stehen jeweils das im Rechnerknoten lokal installierten Betriebssystem und die auf dem Rechner verfügbaren Transportdienste sowie Netzbetriebssystem zur Verfügung. Das verteilte System DCE steht im Kern des Schichtenmodells. Das oberste Element, die verteilte Anwendung, greift generell direkt oder indirekt auf alle anderen Komponenten. Das bedeutet, daß höhere Komponenten direkt und auf die untere Komponenten durchgreifen können. Nach Einsatzmethode in den Anwendungen unterscheiden sich zwei artige Dienste, Basisdienste und weitergehende Dienste, im verteilten System DCE. Threads, RPC, Cell Directory, Security und Distributed Time Service sind als Basisdienste, deren Programmschnittstelle dem Anwender bekannt sein müssen. Dadurch können sie in den verteilten Anwendungsprogrammen direkt eingesetzt werden. Die anderen Dienste, der Global Directory Service, das Distributed File System, der Diskless Support Service und die PC-Integration werden als weitergehende Dienste bezeichnet. Sie werden dem Anwender vielmehr direkt an der konventionellen Betriebssystemschnittstellen angeboten.

9.2.2 Struktur von DCE-Cells

Eine DCE-Umgebung wird wegen organisatorischen Gründen in den sogenannten Cells aufgeteilt. Wie Bild 9.2 gezeigt besteht jedes Cell aus einer Menge von Rechnerknoten und ist strikt disjunkt mit anderen Cells. Außerdem wird jedes Cell getrennt und weitergehend autonom verwaltet. Dadurch werden große Systeme besser beherrscht. Cells dienen einerseits zur effizienten Realisierung der DCE-Dienste innerhalb einem Cell, andererseits zur Reduzierung der Anzahl der erforderlichen Interaktionspfade zwischen DCE-Komponenten auf verschiedenen Rechnern. Zum Beispiel kann Produktmanagement meistens innerhalb Cell A (9.2) geleistet werden.

9.2.3 Die Teilkomponenten des DCE

Thread Service

Der DCE Thread Service realisiert eine portable Implementierung von leichtgewichtigen Prozessen (Threads) gemäß dem Standard POSIX 1003.4a. Wie im Bild 9.3 angedeutet unterscheiden sich Threads (Prozesse) von Betriebssystemprozessen dadurch, daß sie auf gemeinsamen Adressraum zugreifen können. Threads dienen zur parallelen Ausführung verschiedener Sektionen eines Programmes, aber jede Thread hat eigenes aus PC, SP und Stack bestehendes Prozeßkontext. Sie werden

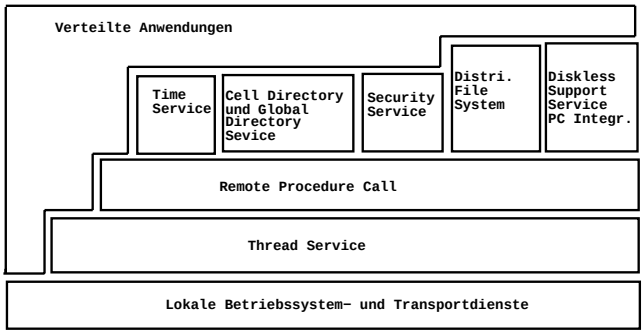


Abbildung 9.2: Gesamtarchitektur der DCE

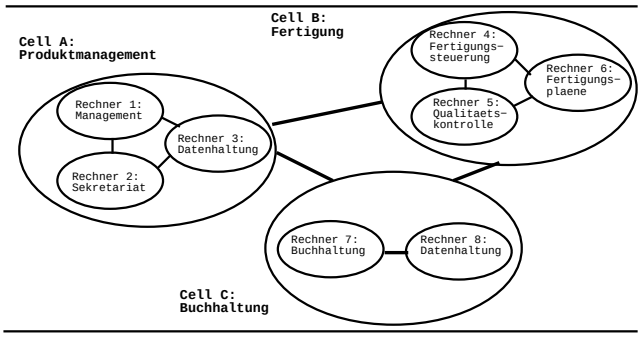


Abbildung 9.3: Kooperierende DCE-Cells

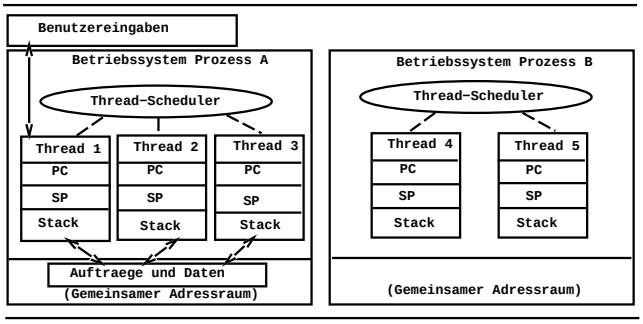


Abbildung 9.4: Struktur von Threads

von einem speziellen Thread-Scheduler verwaltet. Übrigens verfügt das DCE Bibliotheksfunktionen zum Erzeugen und zum Löschen von Threads sowie zur Thread-Synchronisation. Ein Beispiel für die Parallelität ist im Bild 9.4 (links) dargestellt, während Thread 1 ständig auf Benutzereingaben wartet, führen die anderen zwei Threads die Berechnungsaufträge aus.

Remote Procedure Call

Der RPC ist der Basiskommunikationsmechanismus im DCE. Er realisiert Fernaufrufe im Rahmen des Client/Server-Modells. Ein entfernter Prozeduraufruf läuft grob wie folgend ab (siehe Bild 9.5):

1. Binden: Ein Client wählt mit Hilfe der Schnittstellenspezifikation den gewünschten Server aus.

2. Aufrufübertragung: Der Client setzt lokal einen Aufruf ab und dieser wird von System in Form von Nachrichten an den Server übertragen.

3. Aufrufausführung: Das Laufzeitsystem des Servers dekodiert den empfangenen Nachrichten und selektiert die aufzurufende Prozedur. Danach wird die Prozedur ausgeführt.

4. Optionale Unterbeauftragung: Während der Ausführung kann geschachtelter Fernaufruf von Prozedur (in Server 2) abgesetzt werden.

5. Ergebnisrückgabe: Der Server kodiert die Ergebnisse des Aufrufs und überträgt diese an den rufenden Client zurück. Danach setzt der Client seine Bearbeitung fort.

Das DCE RPC besteht aus mehreren Teilkomponenten, die jeweils verschiedene Funktionalität dargestellt. Die Zusammenwirkung zeichnet sich im Bild 9.6 aus. Die Beschreibungssprache IDL (Interface Definition Languge) spezifiziert RPC-Schnittstellen, die mit dem IDL Compiler übersetzt werden. Der UUID Genarator des DCE RPC erzeugt systemweit eindeutige Kennungen (UUIDs, Universal Unique Identifiers) zur Identifikation der RPC-Schnittstellen. In jedem RPC-fähigen Rechnerknoten dient ein RPC-Dämon während des Bindevorgangs zur Zuordnung ankommender Aufrufe. Die Abwicklung von RPCs zur Laufzeit erfolgt durch RPC-Laufzeitsystem. Außerdem verfügt RPC noch einen Kontrollprogramm (rpccp, RPC Control Program) für die Verwaltungsaufgaben.

Directory Service

In einem verteilten System dient Directory Service zur Abbildung logischer Namen (Benennung von Instanzen, wie z.B Modulen, Endbenutzern, ...) auf Netzadressen. Damit kann die Adresse von Instanz mittels einem Directory Service bei Angabe des logischen Namens geliefert werden. Der setzt sich wieder aus zwei Teilkomponenten, nämlich Cell Directory Service (CDS) und Global Directory Service (GDS) zusammen.

Der CDS verwaltet Namen innerhalb einer DCE-Cell und besteht aus einem oder mehreren CDS-Servern. Dadurch wird die Effizienz bei Namensanfragen erheblich erhöht. Wegen Replikation steigt auch die Verfügbarkeit des CDS. Die Namensverwaltung im CDS (Bild 9.7) ist wie im UNIX über einen hierachischen Namensraum. Der GDS ist für die Abbildung von Namen, die über die Grenzen einzelnes Cell hinaus verwaltet sind, zuständig. Damit stellt Der GDS das Bindeglied zwischen verschiedenen CDS dar. Das heißt, der GDS stellt den Namen von Cells im globalen Namensraum dar, zum Beispiel:

/../C=DE/O=OSF/OU=Entwicklung/Abteilung1
/WS5/printserver

Der GDS wird wie CDS auch durch mehrere Server implementiert. Hier werden Replikationstechniken (Serverseite) zur Erhöhung der Verfügbarkeit und Caching-Mechanismen (Clientseite) zur Effizienzsteigerung eingesetzt.

Security Service

Durch *Authentisierung*, *Autorisierung* und *Verschlüsselung* kontrolliert der DEC Sicherheitsdienst den Zugriff unberechtigter Benutzer oder Programme auf Daten, Dienste und Nachrichten innerhalb DCE-Umgebung. Dieser Dienst wird primär zusammen mit dem RPC eingesetzt. Weiterhin bietet DCE Security Service nach Granularität der Authentisierung fünf verschiedene Sicherheitsklassen an, nämlich:

Klasse 1 : Authentisierung zu Beginn einer Interaktionsphase mit einem Server.

Klasse 2 : Authentisierung pro Aufruf eines Servers.

Klasse 3 : Authentisierung pro Übertragungspaket.

Klasse 4 : Schutz vor Modifikation von Nachrichten.

Klasse 5 : Vollständige Verschlüsselung.

Der grundlegende Ablauf bei der Anwendung des Security Service durch einen Benutzer ist im Bild 9.9 dargestellt. Dieser wird in vier verschiedenen Phasen gegliedert:

Benutzerverwaltung: Hier trägt Systemmanager einen Benutzer mit seinem Namen und seinem Paßwort, der von Register-Server verwaltet wird, in eine globale Security-Datenbisis des Sicherheitsdienst ein (1).

Login: Die gesamte Anmeldung eines Benutzers setzt sich aus (2), (3) zusammen und wird durch Login-Komponente (Clientseite), Authent.-Server und Security-Datenbasis (Serverseite) unterstützt.

Autorisierung: Diese Phase schließt (4), (5) ein. Der Priviledge Server überprüft die Zugriffsrechte auf einen bestimmten Server mittels Autorisierungsinformation und kodiert bzw. liefert die Autorisierungsinformation in Form eines Priviledge Attribute Certificates (PAC).

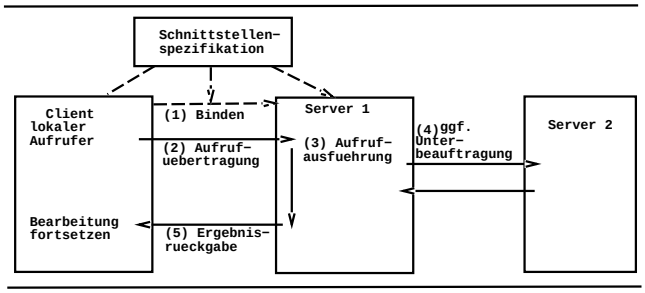


Abbildung 9.5: RPC mit Unterbeauftragung

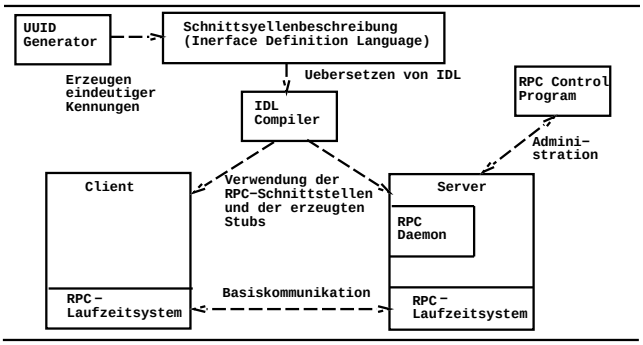


Abbildung 9.6: Teilkomponenten des DCE RPCs

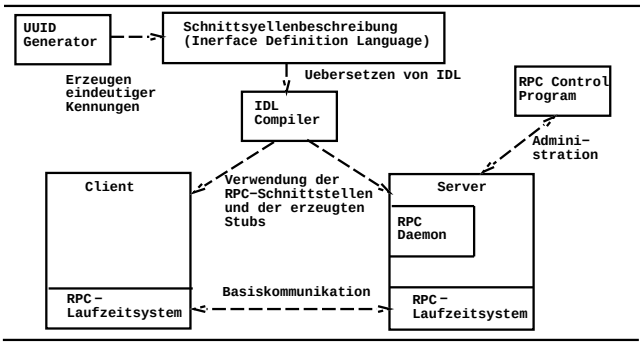


Abbildung 9.7: Teilkomponenten des DCE RPCs

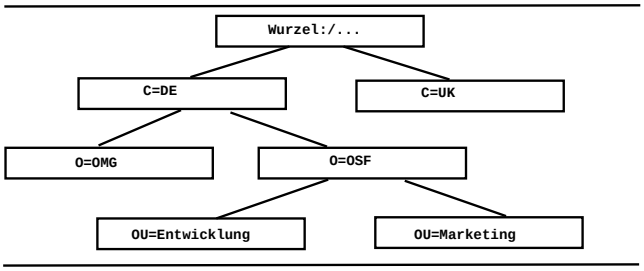


Abbildung 9.8: Beispiel eines einfachen GDS Namensraums

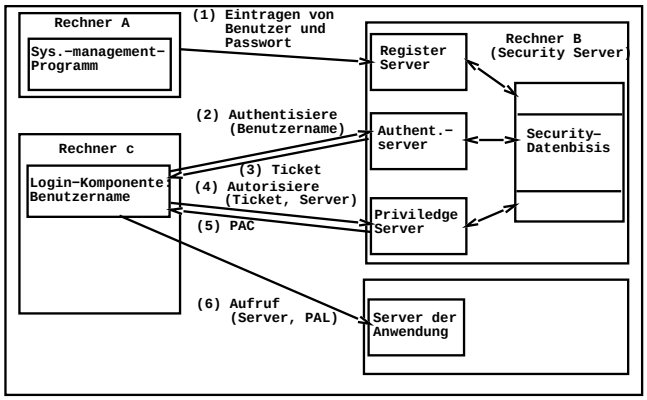


Abbildung 9.9: Authentisierung und Autorisierung

Server-Aufrufer: Der RPC-Aufruf wird hier ausgeführt, zugleich überprüft der Server nochmal die Zugriffsrechte. Wenn sie nicht stimmen, wiederholen sich die Vorgänge (4), (5), (6).

Distributed Time Service

Der DCE Distributed Time Service hat die Aufgabe, im verteilten System die System-Uhren in verschiedenen Rechnerknoten zu synchronisieren. Wie im Bild 9.10 dargestellt, kommen die Synchronisationen hauptsächlich durch periodische Anfragen an Time Server zustande. Innerhalb eines Cell existiert mindestens ein Time Server und die anderen Rechner im Cell bestimmen ihre Systemzeit durch den Time Clerk, der periodisch den Time Server anfragt. Über Cell hinaus dient der Globale Time Server zur Konfiguration von Bereitstellung von Zeitwerten. Außerdem verfügt der DCE DTS über externe Zeitgeber (Time Provider) zur Synchronisation mit exakten Realzeiten.Einerseits wird dem Time Provider über Funk die genaue Zeit versorgt oder manuell eingespeist. Andererseits liefert dieser den Globale Time Servern die genaue Zeit über Time Provide Interface (TPI).

Distributed File System

Das Distributed File System (DFS) dient zur systemweiten verteilten Dateiverwaltung. Das DFS wird ebenfalls wie DTS durch mindestens einen Server in jedem DCE-Cell realisiert und die anderen Rechnerknoten im Cell werden als DFS-Client bezeichnet. Die DFS Server-Komponentenstruktur ist im Bild 9.11 gezeichnet. Dadurch können Anwender so auf Dateien zugreifen, wie sie es von lokalem Dateisystem gewohnt sind.

Jeder DFS-Server besitzt ein lokales Dateisystem, das zur physikalischen Dateiverwaltung auf jeweils einem Rechner dient. Zur verteilten Dateiverwaltung verfügt DFS-Server weitere Teilkomponenten, nämlich Fileset und Fileset-Verwaltung, Dateireplikation, Cache-Synchronisation, Backupverwaltung. Ein Fileset stellt die Einheit der Verteilung beim DFS dar und entspricht einem Teilbaum des Datei-Namensraums. Die Filesetverwaltung bietet Mechanismen zur Lokalisierung eines Fileset beim Zugriff auf eine Datei und Techniken zur Replikation von Filesets. Beim Zugriff auf eine Datei hält sich der entsprechende Client die gesamten Dateidaten oder eine größere Menge von Dateidaten in seinem lokalen Cache, dadurch kann die gesamte Effizienz erheblich erhöht werden. Im Regelfall werden die Daten im Cache erst nach Schließen der Datei zum Server zurückgeschrieben. So entstehen mögliche Konsistenzprobleme bei nebenläufigen Dateizugriffen. Die Cache-Synchronisation wird dafür benutzt,daß alle nebenläufigen Clients über die Änderungen an einer Datei informiert werden und ihre Cache vor dem erneuten Öffnen der Datei aktualisieren.

Diskless Support Service

Der DCE Diskless Support Service (DSS) stellt Dienste dar, die den Betrieb von Workstation ohne Hintergrundspeicher in einer verteilten DCE-umgebung unterstützen. Auf jedem plattenlosen Rechner steht ein Diskless Client zur Verfügung, der, wie im Bild 9.12 gezeigt, durch Kommunizieren mit entsprechenden Servern die normalerweise lokal verfügbare Funktionalität des Hintergrundspeicher erbringt. DSS besteht hauptsächlich aus Boot Server und Swap Server. Der Boot Server dient beim Systemstart zum Laden des Betriebssystemkerns über das Netz. Der Swap Server unterstützt das temporäre Auslagern von Betriebssystemprozessen. Außerdem stellen DFS und Configuration Server einem Diskless Client zur Verfügung, um Verwaltung von Dateien bzw. Initialisierung von Konfigurationsdaten zu ermöglichen.

PC Integration

PC Integration unterstützt Personal Computer unter den Betriebssystemen MS/DOS, OS/2 und UNIX in den Nutzungen der Dienste und Anwendungen in DCE-Umgebung. Auf einem PC, der grundsätzlich als Client arbeitet, stehen die Treads und RPC sowie die anderen auf Client-Seite verfügten DCE-Dienste zur Verfügung. Durch die Client-komponenten auf PC-Seite wird der entfernte Dateizugriff mittels des PC/NFS-Protokolls unterstützt. Die Operationen

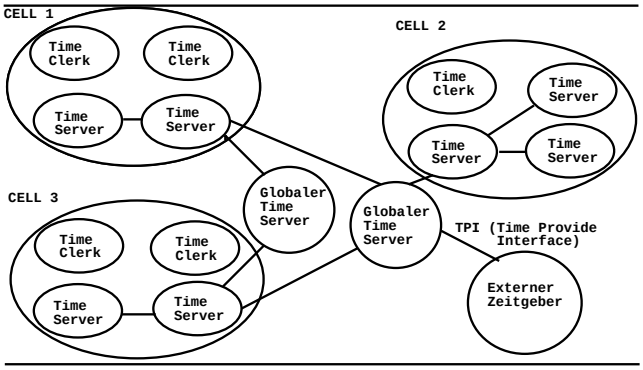


Abbildung 9.10: Time-Server Konfiguration

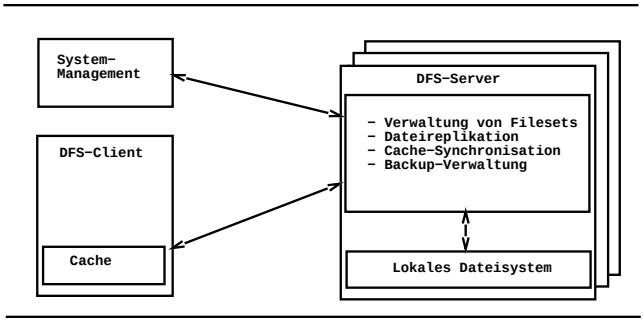


Abbildung 9.11: Struktur des Distributed File Systems

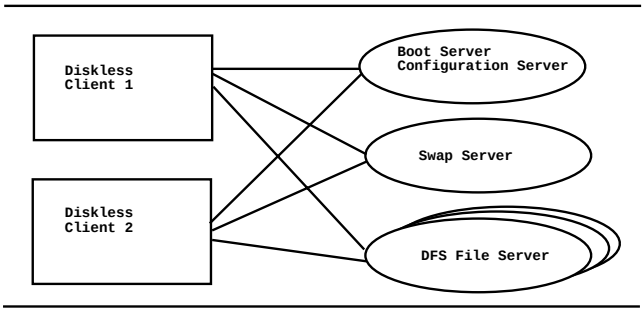


Abbildung 9.12: Diskless client/server Konfiguration

zum Bearbeiten, Kopieren und Migrieren von Dateien in UNIX-Workstation können einwandfrei auf PCs ausgeführt. Übrigens wird die entfernte Nutzung von Druckern durch PCs mittels PC Integration ermöglicht.

9.2.4 Zusammenwirken der DCE-komponenten

Die DCE-Komponenten benutzen sich in vielen Fällen gegenseitig, um ihre verteilte Funktionalität zu erbringen. Diese ist im Bild 9.13 zusammen gefaßt. Als Beispiel behandeln wir hier die gegenseitige Verwendung von RPC und anderen DCE-Service. Der RPC verwendet den Thread Service zur Durchführung nebenläufiger Server-Implementierungen. Weiterhin findet der RPC beim Aufruf mit Hilfe von CDS die Platzierung von entsprechendem Server, sowie verwendet den Security Service, um authentisierte, autorisierte und verschlüsselte Aufrufe zu realisieren. Da der RPC der Kommunikationsmechanismus in DCE-Umgebung ist, interagieren sowohl Clients und Server der verschiedenen Dienste von DTS, CDS, DFS, DSS und PC Integration als auch die Server untereinander mittels RPC. Aber der GDS nutzt ISO/OSI-Kommunikationsprotokolle anstatt des RPC, um eine X.500-standard-konforme Implementierung zu realisieren.

9.3 Literaturverzeichnis

[Sch93]

Diese Komp.: benutzen:	Threads	RPC	CDS	GDS	DTS	Secutity	DFS
Threads		ja	ja		ja	ja	ja
RPC			ja		ja	ja	ja
CDS		ja			ja	ja	ja
GDS			ja				
DTS		ja	ja			ja	ja
Secutity		ja	ja		ja		ja
DFS							

Abbildung 9.13: Zusammenwirken der DCE-Komponenten

Kapitel 10

SNMPv2: Manager-to-Manager MIB und Intermediate-Manager MIB

Frank Abegg

10.1 Einführung

Das "Simple Network Management Protocol" (kurz: SNMP) hat sich für Managementanwendungen in TCP/IP-Netzwerken mittlerweile etabliert. Die zweite Version von SNMP, auch SNMPv2 genannt, bringt viele Verbesserungen gegenüber dem starren und einfachen Konzept der ersten Version. Man findet diese hauptsächlich im Sicherheitsbereich und bei den Kommunikationsmöglichkeiten. In diesem Text sind vor allem letztere von Interesse, insbesondere die Ermöglichung der Kommunikation zwischen Managementanwendungen.

Dazu wurden "Datenbanken", sogenannte MIBs, erstellt, die diese Kommunikation weiter vereinfachen und unterstützen sollen. Zum einen gibt es die sogenannte Manager-to-Manager MIB, die eine ereignisgesteuerte Kommunikation ermöglichen soll und inzwischen fester Bestandteil von SNMPv2 ist, und zum anderen gibt es die sogenannte Intermediate-Manager MIB, die die Verteilung von Skripten vorsieht aber bisher aus dem Entwurfsstadium nicht herausgekommen ist.

In Abschnitt 10.3 und 10.4 sollen die Konzepte der beiden MIBs und ihre Objekte näher erläutert werden. Als nächstes werden aber noch die wichtigsten Begriffe von SNMP erklärt.

10.2 SNMPv2-Netzwerkmanagement

In diesem Abschnitt werden die wichtigsten hier benötigten Begriffe von SNMP bzw. von SNMPv2 erklärt. Außerdem wird auch auf die Skriptsprache von SNMPv2 kurz eingegangen, die insbesondere für das Intermediate-Manager Konzept benötigt wird.

10.2.1 Begriffe

Das SNMP-Konzept umfaßt vier Hauptkomponenten, die ein anspruchsvolles und weitgehend automatisiertes Management für große Netzwerke erlauben. Es wird hauptsächlich für TCP/IP-Netzwerke verwendet und hat sich hier auch durchgesetzt. Die zweite Version hat sich aufgrund ihrer höheren Komplexität noch nicht so sehr durchgesetzt, bietet aber mehr Komfortabilität und mehr Sicherheit. Nichtsdestotrotz soll sie in diesem Text als Grundlage dienen.

Die erwähnten vier Hauptkomponenten sind:

1. die SMI (Structure of Management Information: sie bildet die Grundlage für die Beschreibung und Benennung von Objekten und der MIBs (siehe 2.). Sie erfolgt in der Abstract Syntax Notation One (kurz ASN.1).

2. die MIB-II ist die ,,Standard-Datenbank" aller gemanagten Objekte des Netzwerks. MIB heißt ,,Management Information Base". MIBs sind Ansammlungen verwandter Objekte und bilden immer die Grundlage eines TCP/IP-Netzwerkmanagementsystems.

3. der RFC 1445: In ihm sind einige Verwaltungs- und Architekturdefinitionen enthalten.

4. das Protokoll selbst, über das die Informationsübertragung zwischen Managementanwendung und Managementdatenverarbeitungseinheit festgelegt wird.

Mit Hilfe dieser vier Hauptkomponenten will man nun die Elemente im Netzwerk steuern und überwachen. Dies sind in der Regel alle ans Netz angeschlossenen Hostrechner, Geräte (logische und physikalische), Router und ähnliches. SNMP-Netzwerkmanagement erfolgt im wesentlichen mit den drei nachfolgend aufgeführten Netzwerkelementen:

Erläuterung zu Abbildung 10.1 :

(a) SNMPv2 Netzwerkmanagement:

Dicker Pfeil Kommunikation zwischen zwei Managmentstations,

A Agent,

1,2 : SNMP-Skripte.

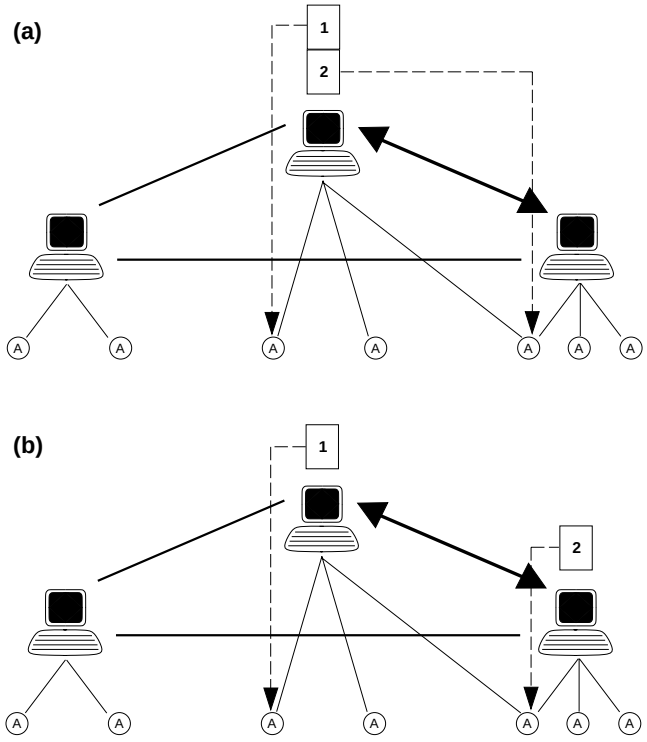


Abbildung 10.1: SNMPv2 Netzwerkmanagement

(b) Verteiltes Netzwerkmanagment durch Herunterladen von Skripten:
Die Skripte 1 und 2 werden nun von zwei verschiedenen Managementstations ausgeführt. Ein Herunterladen auf einen Agent ist auch möglich.

1. Managementstations (MS): Auf ihnen werden die Netzwerkmanagmentanwendungen gefahren, das heißt sie bilden die direkte Schnittstelle zum Netzwerkmanager. Aus der Sicht des Client-Server-Betriebs bilden sie die Clients: Sie fordern die Managementdaten an.

2. Managementagents (MA): Diese verarbeiten direkt die Daten eines Gerätes und sind somit die „Server“ für die Managementstations. Man bezeichnet sie daher auch als Verarbeitungseinheiten.

3. Duale-Role-Stations (DRS) sind gleichzeitig Managementstation und Managementagent, das heißt sie verarbeiten gleichzeitig Gerätedaten und fahren Managementanwendungen, die andere Managementstations bedienen. Zur Kommunikation zwischen zwei Stations stellt SNMPv2 standardmäßig den „inform_request“-Befehl zur Verfügung.

In Abbildung 10.1 sind die SNMPv2-Managementarten noch einmal graphisch dargestellt.

Alle Netzwerkelemente bzw. gemanagten Objekte können zu Gruppen, den sogenannten Partys, zusammengefaßt werden, womit automatisch gewisse Sicherheitsaspekte abgedeckt sind.

Weiterhin sind noch der sogenannte MIB-View und die SNMPv2-Kontexte von Bedeutung. Ersterer ist immer eine Untermenge aller gemanagten Objekte einer Station oder eines Agents. Letztere stellen die managebaren Objektressourcen beim Zugriff auf ein Objekt dar.

Im folgenden Abschnitt wird näher auf die Skriptsprache, die SNMPv2 zur Verfügung stellt, eingangen.

10.2.2 Die SNMPv2-Skriptsprache

Die Skriptsprache von SNMPv2 dient der Automatisierung und Strukturierung von SNMPv2-Anfragen, das heißt man kann so anspruchsvollere und komplexere Netzwerkmanagementanwendungen fahren. Sie stellt also eine Schnittstelle für den Netzwerkmanager dar, die recht einfach aufgebaut ist, denn die meisten Konzepte sind aus höheren imperativen Programmiersprachen bekannt.

Nachfolgend sind die wichtigsten Elemente aufgezählt:

Variablen definieren wie gewohnt Platzhalter für Speicherplätze (sogenannte „Varbinds“), die nicht zwingend typgebunden sind. Man nennt Variablen auch Objekte, weil sie instanziiert werden und sich aus den drei Feldern `Name`, `ID` und `Wert` zusammensetzen.

Typen gibt es auch, aber nur einfache Datentypen wie Integer und Counter32 und eine Verkettung dieser.

Operationen mit Werten und Variablen bilden dann wie üblich **Ausdrücke**. Der Ergebnistyp einer Operation entspricht dabei immer dem Typ des Operanden links vom Operator.

Anfragen unterscheidet man von den anderen Anweisungen, da sie Funktionen ähnlich sind und zudem die zentralen Elemente von SNMP bzw. SNMPv2 bilden. Außerdem gibt es eine Vielfalt von Anfragearten, hauptsächlich unterscheidet man das Abfragen (get_request) und die Modifikation (set_request) von Objekten. Managment-Stations können über die `inform_request`-Anweisung Daten austauschen. Desweiteren kann man die Anfragen synchron oder asynchron absetzen.

Anfrage-Handler behandeln im Bedarfsfall Anfragefehler derart, daß bestimmte Meldungen gesendet werden.

An Kontrollflußelementen gibt es die ,,if-then-else"-Bedingung und die ,,while"-Schleife. Letztere funktioniert wie üblich oder auch wie eine ,,repeat-until"-Schleife.

Anweisungen beenden ein Skript (mit `return`), drucken auf dem Bildschirm (`print`), starten einen Systemaufruf (`exec`) usw.

Funktionen komplettieren die Sprachelemente soweit. Man unterteilt sie in `interne` und `externe` Funktionen: Erstere sind fest vorhanden, während letztere beliebig definiert werden.

Mit diesen Elementen können nun komplexe SNMPv2-Skripte geschrieben und neue MIBs definiert werden. Eine mittlerweile schon fest etablierte MIB ist die Manager-to-Manager MIB, die im nächsten Abschnitt vorgestellt wird.

Abbildung 10.2 zeigt ein Beispielskript. Die erste Textzeile stellt eine asynchron abgesetzte Anfrage dar: Der Wert eines Objekt namens `sysName.0` wird hier abgefragt. Damit das irgendwann hoffentlich bereitstehende Ergebnis der Anfrage wieder korrekt zugeordnet werden kann, wird eine Identifikationsnummer in der Variablen `requid` abgelegt. Später wird dann das Ergebnis mit der Funktion `receive` erfragt und bei Vorhandensein in der Variablen result abgelegt. Die nachfolgenden Zeilen sorgen dann für eine Fehlerbehandlung und sind eher symbolisch zu verstehen. Die Variable `SNMP_REQUEST_PENDING` ist gesetzt, wenn die Anfrage noch unterwegs ist.

```
...
VAL(requid[0]) = send get_request({sysName.0});
...
result = receive VAL(requid[0]);
if (length(error_list) > 0)
   { # Ein Fehler ist aufgetreten
     if not SNMP_REQUEST_PENDING
        { # Request fehlgeschlagen
          print [``Fehler!''];
        }
   }
else
   { # Es gibt eine Antwort }
...
```

Abbildung 10.2: Beispiel eines SNMPv2-Skriptes.

10.3 Die Manager-to-Manager MIB

Die Kommunikation über `inform_request`-PDUs (`Protocol Data Units`) zwischen zwei Managementstations gibt es erst ab Version 2 von SNMP. Mit ihr wurde dann auch der Versuch gemacht, neue MIBs zu definieren, um das Netzwerkmanagement zu vereinfachen, zu automatisieren und vor allem zu verteilen. Eine davon ist die Manager-to-Manager MIB.

10.3.1 Einführung

Die Manager-to-Manager MIB definiert drei Tabellen für die ereignisgesteuerte Kommunikation zwischen mehreren Managementstations oder zwischen Managementstations und Duale-Role-Stations. Über die Alarmtabelle werden Variablen bzw. zu managende Objekte periodisch und automatisch abgefragt und mit Schwellwerten verglichen. Im ,,Überschreitungsfall" werden bestimmte, in der Ereignistabelle definierte Ereignisse ausgelöst, eventuell zusammen mit dazugehörigen Meldungen aus der Meldungsgruppe.

Einen näheren Einblick in die Funktionsweise erhält man im nächsten Abschnitt bei der Besprechung der einzelnen Tabellen und ihrer Felder.

10.3.2 Objektübersicht

Abbildung 10.3 zeigt die Struktur der Alarmgruppe Manager-to-Manager MIB, deren Haupteil die Alarmtabelle ist. Die Ereignisgruppe und die Meldungsgruppe findet man in Abbildung 10.4.

Zur Alarmgruppe gehört nicht nur die **Alarmtabelle** (`snmpAlarmTable`), sondern auch die Variable `snmpAlarmNextIndex`, die auf den Index der nächsten freien Zeile in der Alarmtabelle verweist (siehe auch Abbildung 10.3).

Eine Zeile in der Alarmtabelle (`snmpAlarmEntry`) kann man als Datensatz mit 12 Datenfeldern auffassen. Im ersten Feld steht der Zeilenindex, im zweiten der Name der abzufragenden Variablen und im dritten die Abfrageperiode, das heißt die Zeit nach der erneut eine Stichprobe der Variablen genommen wird. Ob der absolute Wert abgefragt wird oder ein Differenzverfahren zum Einsatz kommt, wird in Feld 4 (`snmpAlarmSampleType`) festgelegt, während im Feld `snmpAlarmValue` nach einer erfolgreichen Abfrage der Wert der Variblen abgelegt wird.

In Feld 7 und 8 sind nun der obere und der untere Schwellwert, mit dem der Variablenwert

verglichen wird, festgelegt. Überschreitet bzw. unterschreitet nun der Variablenwert einen dieser Werte, so wird ein Ereignis entsprechend dem Eintrag im Feld `snmpAlarmRisingThreshold` bzw. `snmpAlarmFallingThreshold` generiert. Dort steht der Zeilenindex bezüglich der Ereignistabelle, unter welchem die Ereignisbeschreibung abgelegt ist. Tritt schon bei der ersten Abfrage ein Überschreiten auf, so wird ein anderes Ereignis ausgelöst, nämlich dasjenige mit dem Index aus Feld 6 (`snmpAlarmStartupAlarm`). Ähnllich wird bei einer Nichtverfügbarkeit der Variablen das Ereignis mit dem Index aus dem Feld `snmpAlarmUnavailableEventIndex` ausgelöst.

Das letzte Feld einer Zeile bestimmt den Zustand eines Zeileneintrags. Mit ihm kann man neue Einträge kreiieren ('createAndWait'), einen Eintrag aktivieren ('active'), löschen ('destroy') usw.

Die Ereignisgruppe enthält nun die wichtigsten Ereignisdefinitionen und die Meldungstyptabelle (`snmpNotifyTable`) sowie einige einfache Variablen (siehe Abbildung 10.5). Zu letzteren gehören neben `snmpEventNextIndex` auch die Variable `snmpEventNotifyMinInterval`, welche die minimale Wiederholungszeit angibt, und die Variable `snmpEventNotifyMaxRetransmissions`, welche die maximal zulässige Anzahl von Wiederholungen der Eregnisauslösung angibt.

Ein Eintrag bzw. eine Zeile in der Ereignistabelle besteht aus 6 Datenfeldern, wovon das erste wieder den Zeilenindex und das letzte wieder den Zustand der Zeile bestimmt. Im Feld 2 findet man eine ID-Nummer für das Ereignis und in Feld 3 (`snmpEventDescription`) einen Kommentar. Im Feld `snmpEventEvents` wird mitgezählt, wie oft das Ereignis schon ausgelöst wurde, und im Feld `snmpEventLastTimeSent` wird schließlich die Systemzeit zur letzten Auslösung gespeichert.

Die **Meldungstyptabelle** bestimmt den Typ des Ereignisses, das heißt, daß hier im Einzelfall konkret, jedoch abhängig von den beiden zuletzt erwähnten Variablen, festgelegt wird, wie oft ein Ereignis übertragen werden soll und nach welcher Dauer erneut.

In der Meldungsgruppe sind drei Standardmeldungen definiert, die bei den entsprechenden Ereignissen generiert werden. Ihre drei Objekte sind in Abbildung 10.4 dargestellt.

Die Alarmtabelle wird nun ständig automatisch abgelaufen, und so werden kontinuierlich Stichproben der Variablen aus den aktiven Zeilen genommen und mit den Schwellwerten verglichen. Im Überschreitungsfall werden dann die entsprechenden Ereignisse generiert.

Der Ansatz der Manager-to-Manager MIB für verteiltes Netzwerkmanagement basiert nun auf den Anfragen, die SNMPv2 schon bereitstellt, (insbesondere „inform_request“). Ein weiterer Ansatz über eine andere MIB und mit dem Konzept des „Herunterladens“ von Skripten, der sogenannte Mid-Level-Manager-Ansatz, wird nachfolgend beschrieben.

10.4 Die Intermediate-Manager-MIB

Die Intermediate-Manager-MIB dient der Konfiguration und Steuerung einer Dual-Role-Station, wobei die Verteilung des Netzwerkmanagements durch das Verteilen bzw. „Herunterladen“ von SNMPv2-Skripten erreicht wird.

10.4.1 Einführung

Das Prinzip der Verteilung des Netzwerkmanagement über das „Herunterladen“ von Skripten ist in der obigen Abbildung 10.1(b) dargestellt. Die beiden ursprünglich auf der gleichen Station laufenden Skripten werden nun dadurch auf zwei Stations verteilt, daß Skript 2 auf eine andere als die ursprüngliche Station heruntergeladen wird. Die Ergebnisse werden dann wie gehabt über die „inform_request“-Anfrage zwischen den Stations ausgetauscht.

In diesem Fall leistet eine MS also Dienste für eine andere MS, und eine solche in doppelter Rolle (MS und MA zugleich) agierende DRS wird dann auch als Mid-Level-Manager (MLM) oder Intermediate-Manager bezeichnet.

Ähnlich wie bei der Manager-to-Manager MIB werden auch hier wieder drei Tabellen definiert, um einen solchen Mid-Level-Manager einzurichten und zu steuern Eine Übersicht über die Objekte und über das Funktionsprinzip folgt im nächsten Abschnitt.

Eine Anmerkung zur Sicherheit darf hier auch nicht fehlen, da eventuell von einer MS über die Zwischenstelle eines MLM unauthorisierte Zugriffe auf Managementvariablen stattfinden könnten. Um diese zu unterbinden, wäre es angebracht, die Sicherheitsvorrichtungen von SNMPv2 einzubinden.

10.4.2 Objektübersicht

Eine komplette Übersicht über die Intermediate-Manager MIB gibt Abbildung 10.6 wieder. Hier treten auch wieder deutlich die drei Gruppen um die drei Tabellen herum hervor. Als erstes findet man die Kompiliertabelle vor, über die entschieden wird, welcher MLM wann und wie oft ein Skript ausführt. In der Skripttabelle findet man den Code eines Skriptes und die Ergebnistabelle nimmt schließlich die Werte auf, die ein Skript nach Beendigung zurückgibt. Im folgenden soll nun näher auf die einzelnen Objekte der Intermediate-Manager MIB eingegangen werden.

Die Kompiliergruppe besteht aus der Variablen `mlmLock`, die auf den Index der nächsten freien Zeile verweist, der Variablen `mlmScriptwrite`, die festlegt, ob ein Skript nach einer Namensänderung ab-

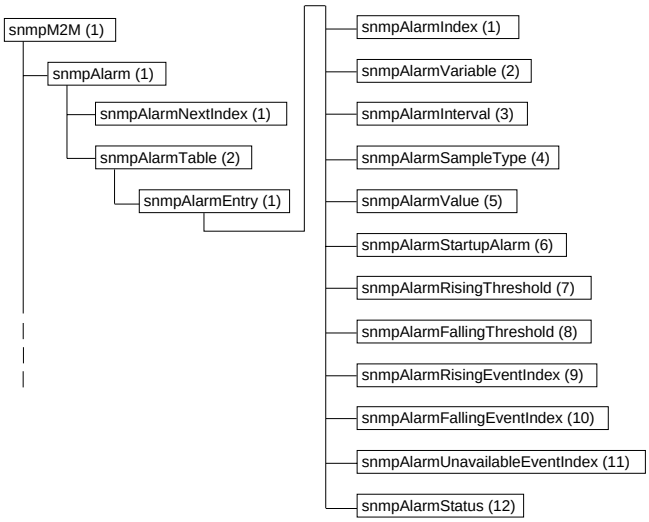


Abbildung 10.3: Manager-to-Manager MIB - Alarmgruppe

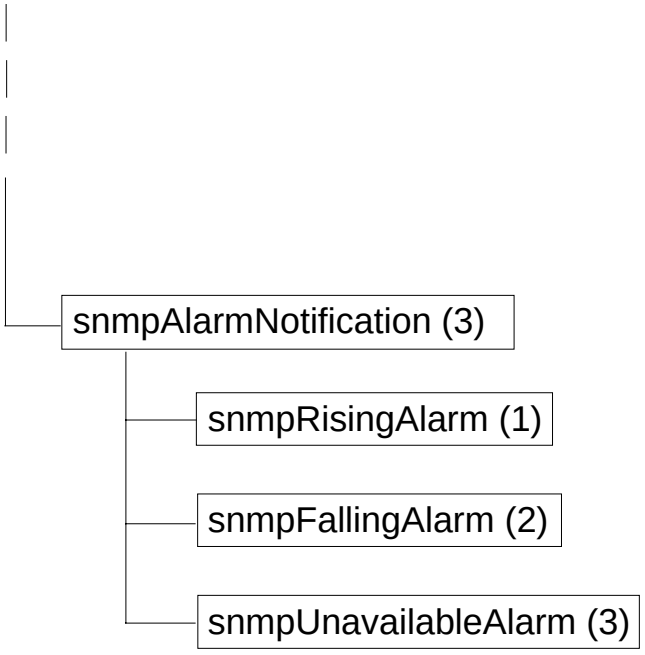


Abbildung 10.4: Manager-to-Manager MIB - Meldungsgruppe

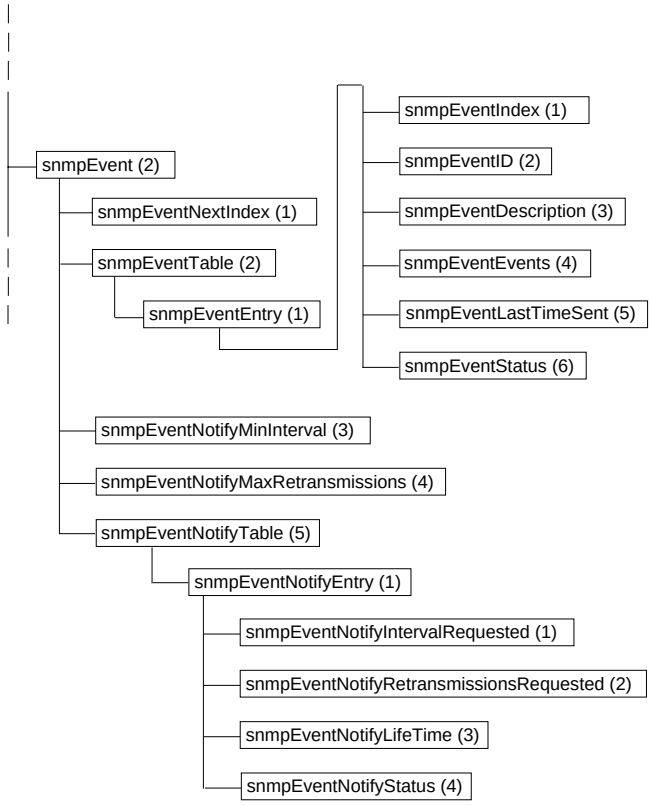


Abbildung 10.5: Manager-to-Manager MIB - Ereignisgruppe

gespeichert werden soll, und aus der Kompiliertabelle `mlmCompileTable`. Weiterhin kann man mit den beiden Variablen `mlmExececutionSpeed` und `mlmTimeSlice` auf die Ausführungsgeschwindigkeit der Skripten Einfluß nehmen. Mit ersterer erhöht man die Auszeit, während letztere bestimmt, wie viele Zeilen der Skripten nach einer Auszeit abgearbeitet werden.

Ein Zeileneintrag umfaßt, wie man in der Abbildung sieht, 7 Datenfelder. Das erste Feld gibt den Index der Tabellenzeile an, das nächste verweist mit einer Skriptnummer auf die Skripttabelle (für den Zugriff auf den zugehörigen Skriptcode). Das Feld `mlmScriptName` kann verwendet werden, wenn eine Datei als Skript dienen soll, insbesondere wird es also zum Herunterladen eines Skripts auf einen anderen Rechner benutzt. In der Regel sollte dann das Feld `mlmScriptIndex` leer sein.

Von besonderem Interesse ist das letzte Datenfeld (`mlmRowStatus`), da mit ihm bestimmt wird, ob das Skript, auf das in der zugehörigen Zeile verwiesen wird, kompiliert, ausgeführt oder gelöscht wird. Im letzten Fall, der wieder mit dem Setzen auf `destroy` erreicht wird, wird nicht nur die Zeile in der Kompiliertabelle gelöscht, es verschwinden auch alle zugehörigen Einträge in der Skripttabelle, sofern diese existieren. Mit dem Eintrag `createAndWait` erzeugt man eine neue Zeile in der Kompiliertabelle, während `active` die Ausführung eines Skripts veranlaßt. Dazu muß das Skript natürlich erst kompiliert werden, was man ebenfalls mit dem Eintrag `active` erreicht, wenn der Zustand vorher `notReady` war. Das Ergebnis des Kompilierlaufes wird dann im Datenfeld `mlmCompileResult` gesetzt. Entweder steht hier dann die Anzahl der Fehler oder eine '0', wenn keine Fehler aufgetreten sind.

Argumente, die während des Skriptlaufs als Eingabe dienen sollen, kann man im Feld `mlmExecutionArgs` angeben. Nach welcher Zeit das Skript nach seiner Beendigung wieder gestartet werden soll gibt das Feld `mlmExecutionPeriod` an.

Die Skripttabelle enthält den Code von zu benutzenden Skripten, und zwar je Tabellenzeile eine Codezeile. Das erste von den vier Feldern jeder Zeile gibt diesmal den Index des Skriptes an, das heißt es treten bei mehrzeiligen Skripten mehrmals hintereinander die gleichen Indizes auf. Die Indizes sind aber insgesamt stets in steigender Reihenfolge geordnet und weisen keine Lücken auf.

Um nun die einzelnen Skriptzeilen auseinanderzuhalten, wird im zweiten Feld jeder Tabellenzeile die Zeilennummer der Codezeile bezüglich des Skriptes angegeben. Die Codezeile selbst ist im Feld `mlmScriptCode` enthalten. Im Feld `mlmCompileErrors` steht dann eine kurze Fehlerbeschreibung, falls in dieser Zeile beim Kompilieren ein Fehler gefunden wurde.

In der Ergebnistabelle werden von einem Skript zurückgegebene Variablenwerte gesichert. Das

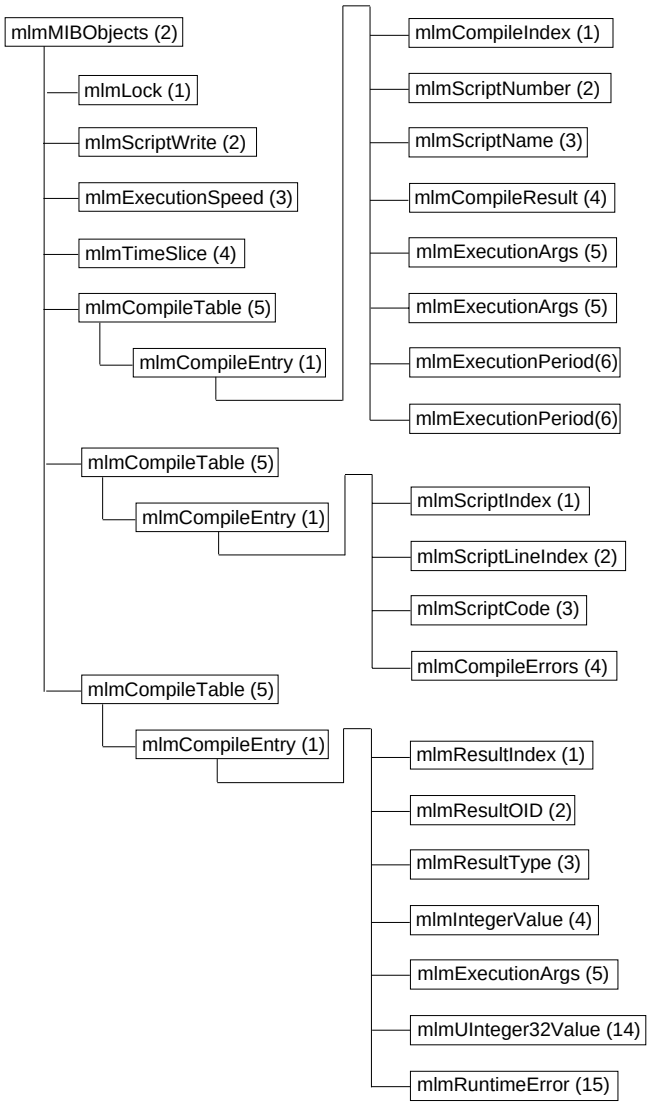


Abbildung 10.6: Intermediate-Manager MIB

Feld `mlmResultindex` zeigt dazu auf die Ergebnisliste des Skripts, während man mit dem Feld `mlmResultOID` das zugehörige Skript wiederfindet. In Feld 4 bis Feld 14 wird dann je nach Typ das Ergebnis eingetragen, wobei der Typ von Feld 3 (`mlmResultType`) geliefert wird. Im letzten Feld werden eventuelle Laufzeitfehler festgehalten.

Eine symbolische Beispieltabellenbelegung und -verkettung der beiden wichtigsten Tabellen zeigt noch einmal Abbildung 10.7. Insbesondere sind die Zeilennummern in den `mlmScriptLineIndex`-Feldern symbolisch zu verstehen. Besonders zu beachten ist der Verweis von der Kompiliertabelle in die Skripttabelle, weil er die Zusammenhänge deutlich macht.

10.5 Zusammenfassung

SNMP hat sich beim Netzwerkmanagement in TCP/IP-Netzen durchgesetzt, nicht zuletzt weil das „OSI-Netzwerkmanagement“ noch nicht über die Entwurfsphase hinausgekommen ist. So hat es bisher auch viele Verbesserungen und Erweiterungen gegeben. Die Verbesserungsvoschläge betrafen hauptsächlich die Sicherheit und die Lockerung des Client-Server-Prinzips, und sie mündeten in der Version 2 von SNMP: SNMPv2. Erweiterungen wurden hauptsächlich über die Definition von zusätzlichen MIBs eingeführt.

Die Manager-to-Manager MIB, die in diesem Text vorgestellt wurde, gehört schon zum festen Bestandteil von SNMPv2, da sie einfach aufgebaut aber dennoch sehr mächtig ist. Sie vereinfacht die Kommunikation zwischen Managementstations, indem die Kommunikation auf die Auslösung und Verarbeitung von Ereignissen zurückgeführt wird.

Die Intermediate-Manager MIB geht hingegen einen anderen Weg: Die Verteilung von Managementanwendungen soll hier durch die Verteilung von Skripten - die Skriptsprache von SNMPv2 wurde in Kapitel 10.2.2 vorgestellt - erreicht werden. Jedoch ist diese MIB bisher nicht aus der Entwurfsphase hinausgekommen, das heißt sie wurde nie implementiert. Dies könnte an der etwas umständlichen und uneinheitlichen Defintion der Tabellen sowie am hohen Speicheraufwand für die Skripttabelle liegen.

10.6 Literaturverzeichnis

[LF89] [Sta93f] [Sta93b]

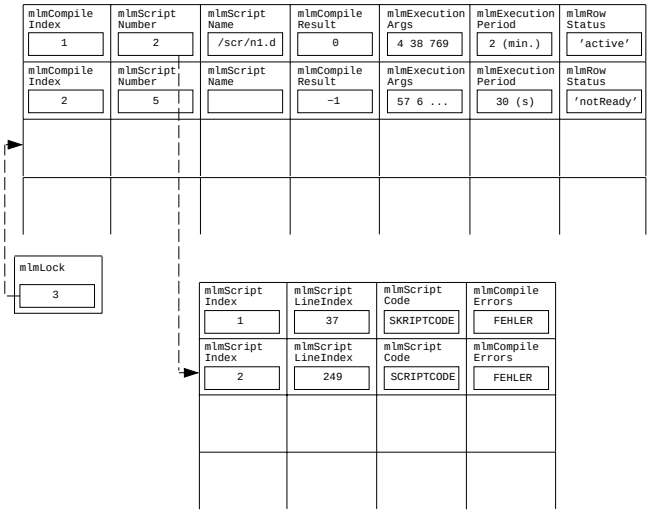


Abbildung 10.7: Beispiel einer Intermediate-Manager-Umgebung mit fiktiven Werten

Kapitel 11

Common Object Request Broker (CORBA)

Juan Uriate

11.1 Kurzfassung

Neue Ansätze in den globalen Netzwerk- und Systemmanagement gehen von einen objektorientierten Modell aus. Es wird hier am Beispiel von CORBA erläutert, welche Vor- und Nachteile solche Managementarchitekturen haben. Anschließend wird ein Blick in die Zukunft geworfen.

11.2 Motivation

Im Moment ist die Konnektivität zwischen heterogenen Systeme gegeben, primitive Dienste, die auf diese einfache Form der Kommunikation aufbauen, Dateitransfer, entferntes einloggen, sind standardisiert, und weit verbreitet. Auf einer höheren Ebene gibt es auch Standards für Remote Procedure Call Systeme, wie z.B.: ONC-RPC oder DCE-RPC. RPC Systeme bilden idealerweise die Bausteine für verteilte Anwendungen, leider gibt es wegen mangelnde Standardisierung auf einer höheren Ebene keine weit verbreiteten verteilten Anwendungen. Ein Netzwerkmanagementstatistik Anwendung, die ihre Arbeit über RPC erledigt, wird sehr unwahrscheinlich mit einer anderen Netzwerkmanagementanwendung von einem anderen Hersteller zusammenarbeiten, obwohl beide womöglich denselben RPC Standard benutzen.

11.3 Object Management Architecture

Um eine herstellerübergreifende Lösung zu diesem Problem im Rahmen einer neuen Technologie zu finden, entstand die Object Management Group (OMG). Es soll auf der Basis des objektorientiertes Modell auf verteilter Basis eine transparente Interoperabilität heterogene Systeme ermöglichen.

Das erstes Produkt dieser Gruppe war das Object Management Architecture (Abbildung 11.1): Die OMA gliedert sich in vier Teile: die Application Objects, die Object Services, die Common Facilities und den Object Request Broker.

Die Object Services bieten die Basisfunktionen zur Verwaltung der Objekte im Netzwerk, insbesondere zur Erzeugung und Verwaltung von Klassen und Instanzen und auch bei Bedarf die Möglichkeit, persistente Objekte zu erzeugen. Vereinfachend kann man sagen, sie bieten die Operationen, die objektorientierte Programmiersprachen lokal anbieten auf eine von der Konzeption heraus verteilte Weise.

Die Common Facilities sind eine Art Bibliothek. Sie sollen Funktionen anbieten, die generellen Charakter haben (z.B.: Drucken oder Email) und dadurch die Anwendungsentwicklung enorm erleichtern und beschleunigen. Es soll möglich sein, durch einfaches ableiten dieser Basisklassen leistungsvolle Anwendungen zu schreiben, ohne jedesmal das Rad neuentwickeln zu müssen. Insbesondere wird sich durch die Standardisierung der Schnittstellen auf diese hohe Ebene ein Softwareteilemarkt eröffnen. Es wird dann möglich sein, wegen die festgelegten Schnittstellen, die verschiedene Teile der OMA von verschieden Hersteller zu kaufen. Diese werden dann problemlos miteinander zusammenarbeiten können.

Die Application Objects sind die eigentliche Anwendungsobjekte, die die Funktionalität der Anwendung realisieren.

Der ORB spielt die zentrale Rolle der OMA (Abbildung 11.2).

Die Aufgabe der ORB ist es, Requests von Objekte transparent über das Netzwerk zu leiten. Transparent in zwei Sinnen: erstens braucht der Aufrufer nicht zu wissen, auf welcher Maschine im Netzwerk sein Kommunikationspartner steht, zweitens muß der Klient nicht unbedingt wissen, auf welchen Maschinentyp oder sogar auf welche Sprachen der Server läuft, die Argumentübersetzung wird automatisch durch den ORB erledigt.

Dazu bietet der ORB eine Reiche von Dienste:

- einen Name Service, der eine eindeutige Benennung der Objekte im Netzwerk ermöglicht.
- einen Request Dispatch Service, der für jeden Request die zugehörige Objekte und Methode(n) findet.

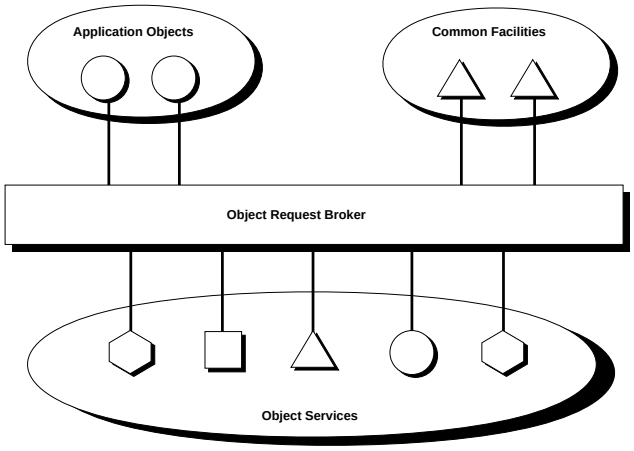


Abbildung 11.1: Object Management Architecture

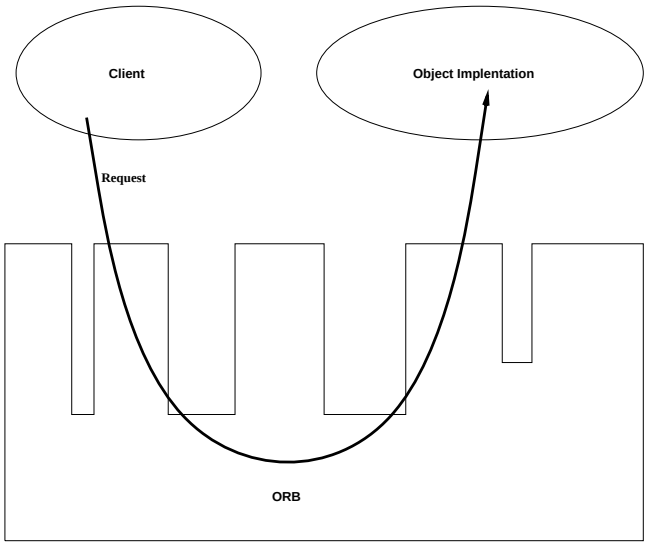


Abbildung 11.2: Versendung eines Requests the Object Request Broker

- das Parameter Encoding, welches, wie bei RPC Systemen, Parameter auf ein maschinen- unabhängiges Format kodiert und dann diese auf das spezifische Maschinen- und Sprachkonventionen zurückdekodiert.
- Delivery Services sind für eine korrekte Zustellung von Requests und Ergebnisse zuständig.
- Activation dient der Verwaltung von persistente Objekte. Vor einen Aufruf muß erstmals das Objekt aktualisiert, und nach dem Aufruf deren Status gesichert werden.
- Exception Handling sind die Dienste, die für die Behandlung von Ausnahmesituationen zuständig sind (Ausfall von einen Knoten, Ausfall von das Netzwerk, Speichermangel, usw.).
- Und die Security Services, die für die Authentifikation von Server und Klient dienen, und für die daran aufbauende Authorisierung des Requests.

11.4 Interface Definition Language

Ein anderer wichtiger Bestandteil von CORBA ist das Interface Definition Language (IDL). Diese Schnittstellenbeschreibungssprache dient für eine zielspracheunabhängige Definition von Attributen, Ausnahmensituationen (exceptions), Typen, Konstanten und Methodenschnittstellen. Es wird dann durch sogenannte Language Mappings auf eine Zielsprache eindeutig abgebildet. Im Moment sind Language Mappings für C definiert, mit der Version 2.0 von CORBA sollte dann das Language Mapping für C++ festgelegt werden. Es sind auch Language Mappings für Ada, Smalltalk und COBOL im kommen.

IDL ist im Grunde genommen C++ mit einigen kosmetischen Änderungen und Erweiterungen, die für die Beschreibung von verteilten Schnittstellen notwendig sind: Es besteht die Möglichkeit, bei der Definition von Methoden, die Parameter als Ein-, Aus- oder Einausgabeparameter zu beschriften. Was in der C++ Welt `class` heißt, wird hier `interface` genannt. Es gibt das neue Schlüsselwort `attribute`: es bezeichnet ein les- und/oder schreibbares Wert. Man kann es als syntactic sugar bezeichnen, denn man hätte im nachfolgenden Beispiel anstatt der letzten Zeile auch folgendes schreiben können:

```
rstate_t get_router_status();
void     set_router_status(in rstate_t state);
```

Beispiel:

```
interface generic_router {
// Typen, die f"ur den Zugriff auf den Router gebraucht werden.

 enum   state_t   { up, down };      // Interface Status
 enum   rstate_t  { up, rebooting }; // Zustand des Routers

 // Eintrag in Routingtabelle
 struct  rtable_entry {
  generic_address dest_addr;         // Zieladresse
  generic_address src_addr;          // Quelladresse
  generic_if_name interface_name;    // Interface
```

```
 state_t         link_state;        // Status des Links
};

// Eintrag in Statistiktabelle
struct  statistic_entry {
 generic_if_name interface_name;    // Name des Interfaces
 counter_t       in_packets;        // Empfangene Pakete
 counter_t       out_packets;       // Gesendete Pakete
 counter_t       err_packets;       // Fehlerhafte Pakete
 time_t          up_time;           // Zeit seit reset
};

// Funktion zum Setzen eines Routingeintrags im Router
error_t SetRoutingEntry {
 in  generic_if_name  interface_name;
 in  generic_address  dest_addr;
 in  generic_address  src_addr;
};

// Funktion zum Lesen eins Routingeintrags f"ur ein Interface
rtable_entry    GetRoutingEntry {
 in  generic_if_name  interface_name;
};

// Lesen der Statistikdaten eines Interfaces
statistic_entry GetStatistic {
 in  generic_if_name  interface_name;
};

// Zur"ucksetzen der Statistikdaten
error_t ResetIfStatistic {
 in  generic_if_name  interface_name;
};

// Lesen/Schreiben des Routerzustandes
attribute rstate_t    router_status;
};
```

Der Kode dieses Beispiels wäre die Beschreibung für eine einfache Schnittstelle zur Verwaltung von Routern, die deren Zustände und Operationen auf diese Zustände charakterisiert. Netzwerkmanagementanwendungen können alle Router, die diese Schnittstelle benutzen, verwalten. Angenommen, Hersteller X will ein Router mit eine spezielle Feature bauen: es soll ein Lüfter haben. Er will aber, daß es weiterhin kompatibel mit den Managementanwendungen bleibt. Die folgende Definition der Schnittstelle würde diese Kompatibilität gewährleisten:

```
interface X_Router: generic_router {
 enum fan_state         { on, off };
 attribute fan_state_t  fan_state;
};
```

Damit kann die alte Netzwerkmanagementanwendung problemlos den neuen Router verwalten, und es ist auch möglich, daß die neuen Eigenschaften des Routers mittels dynamischen Aufrufen benutzt (z.B. durch ein Dialogfenster mit dem Benutzer) werden.

Abb. 11.3 zeigt den IDL Kompilationsprozeß. Dort ist zu erkennen, wie aus die IDL Definitionen folgende Komponenten erzeugt werden :

- Stubs in die Zielsprache zum Mitbinden bei den Klienten. Wie aus der RPC-Welt bekannt sind diese Funktionen nur "hohle" Funktionen. Sie bietet den eigentlichen Schnittstelle zum Klienten.
- Skeletons in die Zielsprache zum Mitbinden bei den Server. Diese Funktionen sind die Callbacks zur eigentlichen Methodenimplementation.
- Persistente Speicherung der Schnittstellendefinitionen im Interface Repository. Das ermöglicht einer Abfrage der Schnittstellen zur Laufzeit.

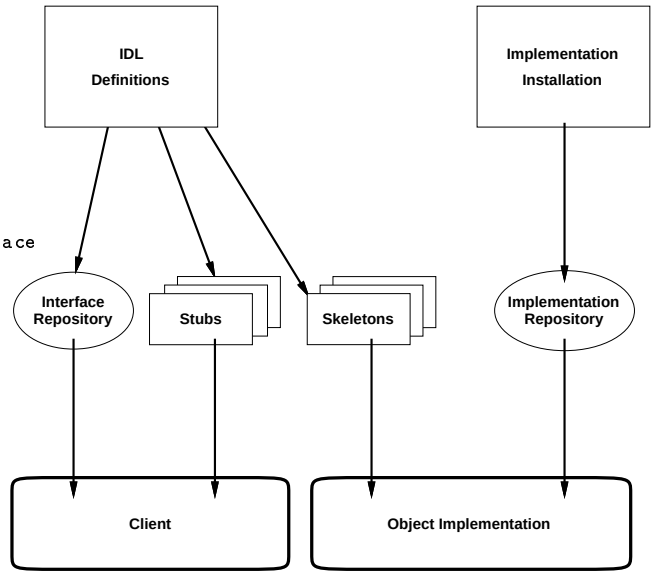


Abbildung 11.3: Interface- und Implementation-Repositories

11.5 CORBA Schnittstellen

Noch ein wichtiger Standardisierungspunkt sind die Schnittstellen zum Object Request Broker (Abb. 11.4). Ein herausragendes Merkmal sind die zwei Arten, Methoden aufzurufen:

- Erstens über die mitkompilierte, statischen Schnittstelle (dunkelgrau in Abb. 11.4), die vom IDL Compiler erzeugt wurde.
- Zweitens, dynamisch zur Laufzeit, durch Schnittstellenanfragen an den Interface Repository (schraffiert im Abb. 11.4).

Die Schnittstelle zum ORB ist auch festgelegt, obwohl die eigentlich internen Schnittstellen von dem ORB zur Außenwelt (schwarz in Abb. 11.4) nicht näher definiert werden. Das Grund dafür ist, daß verschiedene ORB Implementierungen, verschiedene Zugangsmethoden brauchen (eine verteilte objektorientierte Datenbank hat andere Ansprüche und bietet andere Dienstzugangsmethoden als ein einfacher verteilter Broker).

11.6 Basic Object Adapter

Den Basic Object Adapter (BOA) dient primär, Objekte administrativ ins CORBA-Rahmenwerk zu integrieren. Es bietet insbesondere die folgende Funktionen:

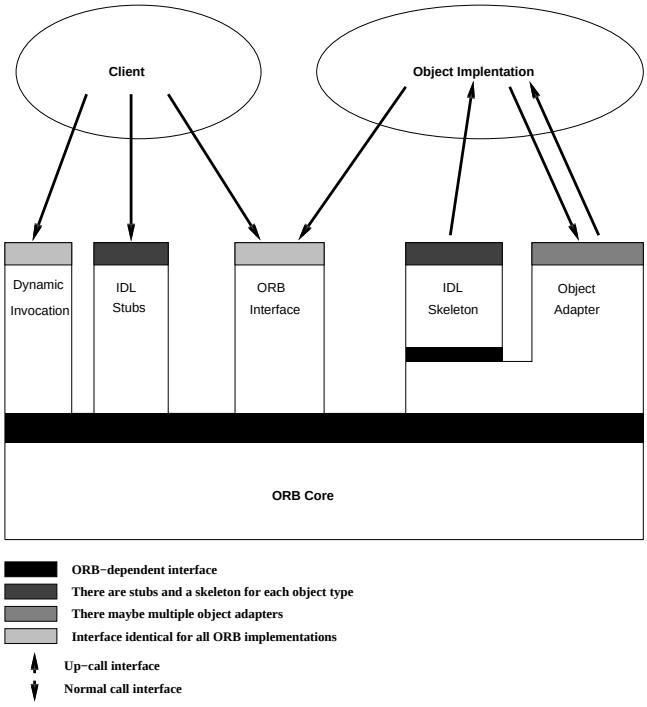


Abbildung 11.4: Struktur der Object Request Broker Schnittstelle

- Erzeugung und Interpretation von eindeutigen Objektreferenzen.
- Identifizierung und Authentisierung der jeweils aufrufenden Client.
- Aktivierung und Deaktivierung der Implementationen. Das spielt eine wichtige Rolle bei der Benutzung von persistente Objekte.
- Und das Wichtigste: den eigentlichen Methodenaufruf mit Hilfe der von IDL-Compiler generierten Rumpfes.

Ein Beispielrequest wäre von Abb. 11.5 abgeleitet wie folgt:

1. Einen Client hat eine Referenz auf ein Objekt, in diesen Fall, ein persistentes Objekt. Die Referenz wird benutzt, d.h. das Clientobjekt ruft eine Methode des referiertes Objektes auf.
2. Durch das ORB wird das Serverobjekt lokalisiert, und den Request and den BOA weitergeleitet.
3. In diesem Fall war das Objektserver nicht aktiv, der BOA muß ihn erstmals aktivieren (Punkt 1 in Abb. 11.5)
4. Das Serverobjekt meldet sich bei der BOA, wenn er bereit ist, Requests für eine bestimmte Schnittstelle zu akzeptieren.
5. Der BOA aktiviert dann die nötige Objektinstanz, die dann bereit ist, die gewünschte Methode auszuführen.

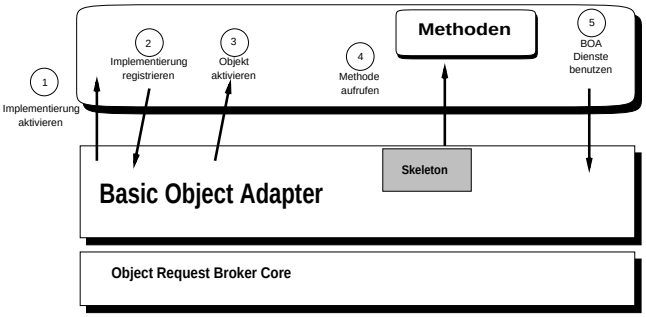


Abbildung 11.5: Struktur und Funktion des Basic Object Adapters

Durch den BOA werden mehrere Formen der Methodenaktivierung unterstützt:

Mehrere Schnittstellen pro Server
Ein einziger Prozeß verwaltet alle Objektinstanzen von einer bestimmte Schnittstelle.

Eine Schnittstelle pro Server
Ein Prozeß betreut alle Instanzen einer Klasse.

Ein Objekt pro Server
Eine Objektinstanz einer Klasse wird durch genau einen Prozeß implementiert.

Mehrere Server pro Objektinstanz
Jede Methode einer Instanz einer Klasse wird durch einen separaten Prozeß realisiert.

11.7 Zukunftsaussichten

CORBA ermöglicht den Wandel von getrennten Homogenen-OS-zentrierten Systemmanagement und protokollzentrierten Netzwerkmanagement zum integrierten heterogenen Netzwerkmanagement. Es wird auch ein Wandel von monolithischen Anwendungen zu komponentenweise zusammengesetzten Anwendungen bewirken. Das wird aber nur dann geschehen, wenn ein Standard für Schnittstelle einer höhere Ebene sich herauskristallisiert, und von vielen Firmen implementiert wird. Ein Bereich, wo dieser Standard schon existiert, ist der des Netzwerkmanagements in Form von dem DME White Paper. Neue Märkte für subapplication Komponenten werden erzeugt. Das wird eine breitere Offenheit im Bereich der Software erzeugen, und eine Senkung der Preise, wenn Firmen gegeneinander in diesen Bereich konkurrieren.

11.8 Literaturverzeichnis

[X/O93] [uRZ93] [Ope94] [Bey93]

Literaturverzeichnis

[Abe93] Sebastian Abeck. SNMPv2. *Praxis der Informationsverarbeitung und Kommunikation*, 16(3):172–172, 1993.

[Bey93] Torsten Beyer. Oberaufsicht: Tivoli management environment: OSF/DME vorweggenommen. *iX Multiuser–Multitasking–Magazin*, pages 58+, August 1993.

[Fra91] Werner Frank. *SNMP eine Einführung.* Schneider&Koch&Co Datensystem GmbH, 1991.

[Gol] Germán Goldszmidt. *On Distributed System Management.* Distributed Computing & Communications Lab, 450 Computer Science Building, Columbia University, NYC, NY 10027.

[Gol93] Germán Goldszmidt. Distributed system management via elastic servers (position paper). In *Proceedings of the IEEE First International Workshop on Systems Management*, April 1993.

[GY92] Germán Goldszmidt and Yechiam Yemini. *Network Management by Delegation (Early Draft).* Distributed Computing & Communications Lab, 450 Computer Science Building, Columbia University, NYC, NY 10027, December 1992.

[GY93a] Germán Goldszmidt and Yechiam Yemini. *Elastic Servers.* Distributed Computing & Communications Lab, 450 Computer Science Building, Columbia University, NYC, NY 10027, September 1993.

[GY93b] Germán Goldszmidt and Yechiam Yemini. Evaluating management decisions via delegation. In *Proceedings of the IFIP International Symposium on Network Management*, Distributed Computing & Communications Lab, 450 Computer Science Building, Columbia University, NYC, NY 10027, April 1993.

[HG94] Mathias Hein and David Griffiths. *SNMP Simple Network Management Protocol Version 2. SNMPv2 in Theorie und Praxis.* International Thomson, 1994.

[JJMD92] John McConnell, Infonetics Research, Inc., Jacqueline Ross, Hughes LAN Systems, Michael Skubisz, Cabletron Systems, Inc., and Dan Simula, SynOptics Communications, Inc. Hub-centered management: A new LAN management paradigm. *The 8th Interoperability Conference – INTEROP*, page G34, October 1992.

[JS93] Rainer Janssen and Wolfgang Schott. *SNMP: Konzepte–Verfahren–Plattformen. Mit SNMPv2.* DATACOM-Fachbuchreihe. DATACOM-Buchverlag, 1993.

[KP93] G. Knight and G. Pavlou. Runtime support for interaction with real resources in the OSIMIS management platform. In *Second Workshop on Networked Systems Management*, Aachen, Germany; Philips Research Laboratories. Folienkopien., April 1993.

[LF89] A. Leinwand and K. Fang. ... In *Network Management: A Practical Perspective*, number ..., pages 1–17, ..., 1989.

[Ope94] Open Software Foundation. *DME Overview*, OSF document number osf-dme-pd-0394-2 edition, March 1994.

[RF91] Ortwin Rose and Albert Fuß. OSI netzwerkmanagement. *DATACOM Special Netzwerk-Management*, June 1991.

[Ros88] Marshall T. Rose. *Verwaltung von TCP/IP–Netzen. Netzverwaltung und das Simple Network Management Protocol (SNMP).* 1988.

[Sch93] Alexander Schill. *DCE - das OSF Distributed Computing Environment.* Springer Verlag, 1993.

[Sei94] Jochen Seitz. *Integration heterogener Netzwerkmanagementarchitekturen.* PhD thesis, Universität Karlsruhe, Institut für Telematik, Fakultät für Informatik, February 1994. Fortschrittberichte VDI, Reihe 10: Informatik/Kommunikationstechnik, Nr. 289.

[Sta93a] William Stallings. CMIS and CMIP. In *SNMP, SNMPv2 and CMIP: The Practical Guide to the Network-Management Standards*, number ISBN 0-201-63331-0, pages 428–472, Addison-Wesley Publishing Company, Inc., 1993.

[Sta93b] William Stallings. Network control. In *SNMP, SNMPv2 and CMIP: The Practical Guide to the Network-Management Standards*, pages 50–63, Addison-Wesley Publishing Company, Inc., 1993.

[Sta93c] William Stallings. The Open Systems Interconnection (OSI) Reference Model. In *SNMP, SNMPv2 and CMIP: The Practical Guide to the Network-Management Standards*, number ISBN 0-201-63331-0, pages 541–558, Addison-Wesley Publishing Company, Inc., 1993.

[Sta93d] William Stallings. OSI Management Information Base. In *SNMP, SNMPv2 and CMIP: The Practical Guide to the Networkmanagement Standards*, pages 388–428. Addison-Wesley Publishing Company, 1993.

[Sta93e] William Stallings. OSI Systems-Management Concepts. In *SNMP, SNMPv2 and CMIP: The Practical Guide to the Network-Management Standards*, number ISBN 0-201-63331-0, pages 371–428, Addison-Wesley Publishing Company, Inc., 1993.

[Sta93f] William Stallings. Overview. In *SNMP, SNMPv2 and CMIP: The Practical Guide to the Network-Management Standards*, pages 1–14, Addison-Wesley Publishing Company, Inc., 1993.

[Sta93g] William Stallings. Systems-Management Functions. In *SNMP, SNMPv2 and CMIP: The Practical Guide to the Network-Management Standards*, number ISBN 0-201-63331-0, pages 473–540, Addison-Wesley Publishing Company, Inc., 1993.

[uRZ93] T. Mowbray und R. Zahavi. Distributed computing with object management. *ConneXions – The Interoperability Report*, 7(12):18–24, December 1993.

[X/O93] X/Open and OMG. *The Common Object Request Broker: Architecture and Specification*, OMG document number 93.12.?, rev. 1.2 edition, December 1993.

[Yem] Yechiam Yemini. Network management. Distributed Computing & Communications Lab, 450 Computer Science Building, Columbia University, NYC, NY 10027.