Proceedings of *Workstations und ihre Anwendungen SIWORK'96*,
Zurich, Switzerland, 1996

# Integriertes Netz- und Systemmanagement mit modularen Agenten

Stephen Heilbronner, Alexander Keller, Bernhard Neumair

Münchner Netzmanagement Team
Institut für Informatik, Ludwig-Maximilians-Universität München
Leopoldstr. 11B, 80802 München, Germany
e-mail: {heilbron|keller|neumair}@informatik.uni-muenchen.de

**Zusammenfassung** Mit der zunehmenden Heterogenität und Komplexität verteilter, kooperativer DV-Versorgungsstrukturen wird auch das integrierte Management dieser Strukturen immer wichtiger. Sollen Managementlösungen flexibel an unterschiedlichste Einsatzszenarien angepaßt werden können, müssen Managementmodule verschiedener Hersteller nahtlos kombinierbar sein. Der Beitrag diskutiert mögliche Architekturen modular aufgebauter Managementagenten und gibt einen Überblick über Erfahrungen, die bei der prototypischen Implementierung gemacht wurden. Er skizziert weiterhin, wie bei der Realisierung der Agenten auch der Existenz mehrerer, konkurrierender Managementarchitekturen Rechnung getragen und damit die Kooperation über diese Architekturen hinweg ermöglicht bzw. verbessert werden kann.

## 1 Motivation

Client-Server-basierte, kooperative DV-Versorgungsstrukturen gewinnen immer mehr an Bedeutung. Der Trend führt weg von der klassischen Orientierung auf einen Großrechner („*Mainframe*"), er geht hin zu einer koordinierten Kooperation verteilter und heterogener HW/SW-Komponenten, die auf offenen Systemen und leistungsfähigen Kommunikationsnetzen basiert. Der Preis dafür ist ein komplexeres technisches Management dieser Systeme, das in heterogener Umgebung nur auf der Basis standardisierter Architekturen effizient zu bewältigen ist. Insbesondere kann man zukünftig nicht mehr wie bisher streng zwischen der Administration des Kommunikationsnetzes (Netzmanagement) und der Administration der Endsysteme (Systemmanagement) und Anwendungen (Anwendungsmanagement) unterscheiden. Man muß vielmehr das Management aller Komponenten dieser Umgebungen zu einem integrierten Management der DV-Infrastruktur zusammenfassen.

Verteilte Systemumgebungen unterscheiden sich extrem, was ihren Aufbau, ihre Größe oder ihre Ausrichtung betrifft. Es kann folglich auch nicht die eine Managementlösung für alle verteilten Systeme geben. Ziel muß es vielmehr sein, einen „Baukasten" von Modulen zu entwerfen, in dem die Teile, die einzelne Problembereiche bearbeiten, so flexibel wie möglich zusammengesetzt werden können, um für jede Umgebung zu einer optimalen Managementlösung zu kommen.

Die *systemübergreifende* Kombination von Modulen, also die Zusammenarbeit sogenannter *Managersysteme* mit *Agenten* verschiedener Hersteller (siehe Abb. 1) wird durch standardisierte Managementarchitekturen [11, 6, 10] ermöglicht. Sie definieren die dazu notwendigen Managementprotokolle, Informationsmodelle und konkrete Information, die zu Managementzwecken auszutauschen ist, die sogenannte Managementinformation. Sollen die Lösungen flexibel an verschiedene Einsatzumgebungen angepaßt werden können, müssen auch innerhalb der Managementsysteme und der Agenten selbst die (Teil-)Module möglichst frei kombinierbar sein. Auf der Seite der Managementsysteme erreicht man dies durch Offenlegung der Architektur und der APIs sogenannter Managementplattformen. Diese Plattformen liefern damit eine gemeinsame Infrastruktur und Ablaufumgebung für Managementapplikationen verschiedener Hersteller.

Komponenten von verteilten Anwendungen sollten z.B. aus Gründen des Lastausgleichs oder aus organisatorischen Erwägungen heraus möglichst flexibel auf verschiedene Systeme verlagert werden können. Auf einem System sind dann häufig eine Vielzahl von Anwendungskomponenten unterschiedlicher Hersteller in ein integriertes Management einzubinden. Die Module, die den Anschluß einer Komponente an das Management liefern („Ressourcen-Module“, „Subagenten“), werden i.a. mit der Komponente selbst bezogen und stammen folglich ebenfalls von verschiedenen Herstellern. Damit sind auch innerhalb der Agenten Schnittstellen („Intra-Agenten-Schnittstellen“) offenzulegen, um die notwendige herstellerübergreifende Koexistenz und Kooperation der Module eines Agenten zu erlauben. Abschnitt 2 erläutert deshalb die in Abb. 1 skizzierte Architektur und die Schnittstellen eines modular aufgebauten, flexibel konfigurierbaren Managementagenten. Der anschließende Abschnitt 3 skizziert dann eine Realisierung dieser Architektur und mehrerer (Ressourcen-)Module bzw. Subagenten, die für ein integriertes Netz-, System- und Anwendungsmanagement nötig sind. Er zeigt damit, wie Agentenmodule für herstellerübergreifendes Management innerhalb eines Systems koexistieren und kooperieren können. Damit ist eine wesentliche Grundlage für flexibel konfigurierbare Managementlösungen gegeben.

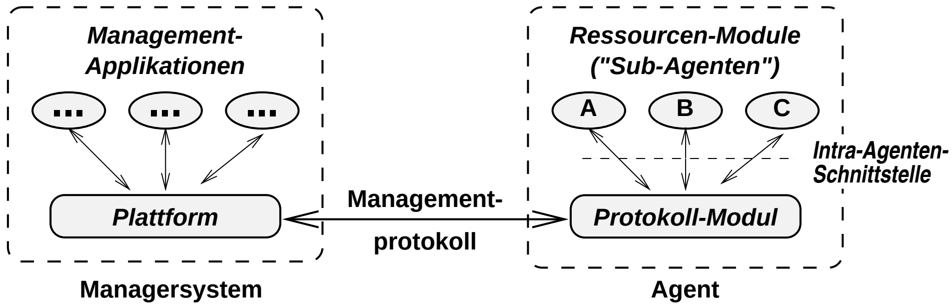


**Abbildung1.** Management-Gesamtarchitektur

Mit der modularen Architektur von Agenten und den erwähnten Plattform-Architekturen sind die notwendigen Voraussetzungen für die Entwicklung integrierter Managementlösungen vorhanden, wenn man nur *genau eine* Managementarchitektur zu berücksichtigen hat. (Damit „sprechen“ dann alle Managersysteme und Agenten das gleiche Managementprotokoll.) In letzter Zeit trat auf dem Weg zum integrierten Management allerdings eine zusätzliche Komplikation auf: neben proprietären Ansätzen wurden *mehrere* Managementarchitekturen standardisiert, die teilweise in Konkurrenz zueinander stehen. Neben dem bekannten, im Bereich des LAN- und Endsystemmanagements weit verbreiteten Internet-Management (*SNMP-Management*) und dem *Desktop Management Interface* der DMTF werden auch die Arbeiten der OMG (*Object Management Architecture, CORBA*) zunehmend beachtet. Diesem Ansatz werden heute gute Chancen auf große Verbreitung für die Entwicklung verteilter Anwendungen eingeräumt. Bewahrheitet sich dies, wird er auch für integriertes Management sehr wichtig werden.

Vermutlich wird sich nicht eine einzige Managementarchitektur durchsetzen; es wird wohl zu einem Nebeneinander von verschiedenen Architekturen kommen. Wirklich integriertes Management setzt also voraus, daß Übergänge geschaffen werden, die eine nahtlose Kombination der Architekturen erlauben. Prinzipiell gibt es hier 3 Alternativen:

- „Multiarchitekturelle Plattform“: der Übergang erfolgt im Managementsystem, die Managementplattform „spricht“ mehrere oder alle „Management-Sprachen“.
- „Management-Gateway“: der Übergang wird durch ein Zwischensystem realisiert
- „Multiarchitektureller Agent“: die notwendigen Abbildungen erfolgen bereits im Agenten (oder werden durch entsprechende Vorgaben an dessen Realisierung überflüssig gemacht)

Abschnitt 4 wird die letzte der drei Alternativen genauer erläutern, da in diesem Beitrag der Schwerpunkt auf den Agenten liegen soll. Er skizziert, wie durch Übersetzung vorhandener Definitionen von Managementinformation in das Informationsmodell einer anderen Architektur und eine entsprechende Realisierung in den Agenten eine Kooperation zwischen diesen Architekturen erreicht werden kann. Er beschreibt einen ersten Ansatz, „Ressourcenmodule“ auch implementierungstechnisch von einer konkreten Architektur unabhängig zu machen und damit eine effiziente Implementierung multiarchitektureller Agenten zu erlauben. Mit derartigen Übergängen ist dann auch in einer Umgebung mit mehreren Managementarchitekturen die Basis für die Entwicklung wirklich integrierter Managementlösungen vorhanden. Ein Ausblick auf noch offene Forschungsfragen auf diesem Weg schließt den Beitrag.

## 2 Aufbau modularer Managementagenten

Mit der zunehmenden Verlagerung der Schwerpunkte des integrierten Managements vom Netzmanagement in Richtung System- und Anwendungsmanagement werden auch innerhalb von Agentensystemen offene Schnittstellen notwendig. Im Gegensatz z.B. zu Routern, deren Komponenten typischerweise vom selben Hersteller stammen, werden Anwendungen, die auf einem Endsystem ausgeführt werden, oft nicht vom gleichen Hersteller wie die Betriebssoftware stammen. „Ressourcen-Module“, die die Einbindung bestimmter Anwendungen in ein integriertes Management erlauben, die also Managementinformation für diese Anwendung bereitstellen[1], wird man i.d.R. vom Hersteller der verteilten Anwendung erhalten. Die agentenseitige Implementierung des Managementprotokolls wird dagegen häufig mit der Betriebssoftware mitgeliefert. Will man verschiedene Ressourcen-Module („Subagenten“) und das Protokollmodul innerhalb des Agenten flexibel und dynamisch zusammenfügen, braucht man natürlich eine standardisierte oder zumindest offengelegte (Programmier-)Schnittstelle („Intra-Agenten-Schnittstelle“, siehe Abb. 1), die z.B. unterstützen muß:

- Registrierung der Ressourcen-Module beim Protokollmodul (mit Angabe, welche Information durch das jeweilige Ressourcen-Modul bereitgestellt wird, d.h. für welche Ressourcen das Modul „zuständig“ ist)
- Übergabe von Aufträgen des Protokollmoduls an die Ressourcen-Module und der Ergebnisse in umgekehrter Richtung
- Übergabe asynchroner Ereignismeldungen von den Ressourcen-Module an das Protokollmodul.

[1] siehe z.B. [12] und [4] mit den darauf aufbauenden RFCs Nr. 1566, 1565, 1611, 1612 und 1697

Mit der Hilfe dieser Intra-Agenten-Schnittstelle sind dann also Agenten zur Laufzeit aus Modulen verschiedener Hersteller konfigurierbar und damit flexibel auf unterschiedliche Einsatzszenarien anpaßbar. Ressourcen-Module für integriertes Management können von den Herstellern dieser Ressourcen mitgeliefert werden und müssen nicht vom Entwickler des Managementsystems nachträglich aufgesetzt werden.

Je nach Managementarchitektur, die durch den Agenten unterstützt werden soll, existieren für die Realisierung der Intra-Agenten-Schnittstelle unterschiedliche Alternativen:

- **OMG Object Management Architecture**: Wird diese Architektur gewählt, besteht ein Ressourcen-Modul aus einer sog. *Object Implementation*. Registrierung von Object Implementations beim *Object Request Broker (ORB)* bzw. Übergabe von Aufträgen oder asynchronen Ereignismeldungen sind Hauptbestandteile der Schnittstelle zwischen ORB und den Objektimplementierungen. Derzeit existieren allerdings noch kaum standardisierte Definitionen von Managementinformation, was Management in heterogener Umgebung noch problematisch macht (siehe auch Abschnitt 4).
- **Internet Management**: Zu dieser Architektur existieren neben einigen herstellerspezifischen Entwicklungen zwei aktuelle standardisierte[2] Schnittstellendefinitionen:
  - **DMTF CI**: Das *Component Interface* innerhalb des *Desktop Management Interface* der DMTF [2] stellt eine Realisierung der Schnittstelle dar, die prinzipiell nicht auf eine spezifische Managementarchitektur zugeschnitten ist, aber erhebliche Ähnlichkeiten mit dem Internet-Management aufweist und damit für diese Architektur am besten geeignet ist. Zur Beschreibung der Managementinformation der Module wird ein eigenes Informationsmodell verwendet, das aber deutlich an das Internet-Informationsmodell angelehnt ist. Trotzdem ist ein Modellwechsel innerhalb der Agenten notwendig.
  - **DPI**: Das Distributed Protocol Interface [13] ist eine Protokollschnittstelle, die in ihrer Struktur auf SNMP und das Informationsmodell des Internet-Management zugeschnitten ist und die obigen Funktionen für diese Architektur realisiert. Datenpakete und -typen können direkt aufeinander abgebildet werden; es ist kein Wechsel des Informationsmodells erforderlich.

Die softwaretechnische Realisierung dieser Schnittstellen geschieht entweder durch Methoden zur Interprozeßkommunikation (wie z.B. *Pipes*) bzw. durch dynamisch ladbare Bibliotheken (*Shared Libraries*, *DLL's*). Letztere haben zwar den Vorteil, zur Laufzeit weniger Ressourcen zu benötigen, ihr Anschluß ist jedoch sehr betriebssystemspezifisch und damit wenig portabel.

Für die erste Realisierung der modularen Managementagenten mit den im folgenden Abschnitt beschriebenen Ressourcen-Modulen wurde die Internet-Managementarchitektur mit der Schnittstelle DPI gewählt, da sich SNMP im LAN-Bereich als Standard durchgesetzt hat und mit DPI keinerlei Modellwechsel innerhalb der Agenten notwendig ist. Abbildung 2 zeigt die Architektur der im folgenden Abschnitt genauer ausgeführten Agenten.

## 3 Komponenten eines modularen Agenten

Im Rahmen der prototypischen Realisierung der modularen Agenten wurde der Schwerpunkt auf das System- und Anwendungsmanagement gelegt. Es wurden also Modu-

[2] Die Arbeiten der vor kurzem gegründeten Gruppe „SNMP Agent Extensibility" der IETF sind noch im Anfangsstadium.

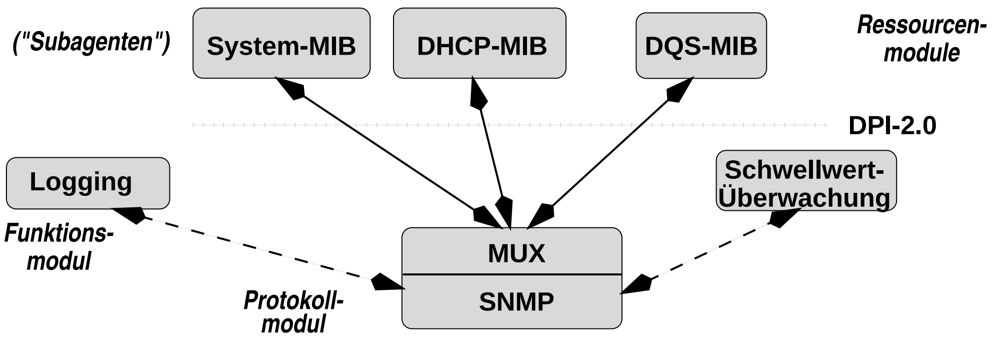


**Abbildung2.** Architektur eines modularen Managementagenten

le für das Management von Endsystemressourcen (siehe 3.1), für verteilte DHCP-Infrastrukturen (3.2) und für das Management von verteilten Batch-Systemen (3.3) entwickelt. Diese Module können, da sie über die Standard-DPI-Schnittstelle mit dem SNMP-Modul kommunizieren, unabhängig voneinander entwickelt werden. Es ist möglich, diese Module (auch mit anderweitig entwickelten Modulen zusammen) dynamisch, also z.B. beim Start der zugehörigen Anwendung, an einen bereits laufenden Agenten anzubinden. Dazu ist keine Unterbrechung eines bestehenden Managementinformationsflusses notwendig.

### 3.1 Management von UNIX-Endsystemen

Das integrierte Management von UNIX-Endsystemen gehört wegen der Vielfalt der Hersteller und Einsatzbereiche sowie der fehlenden Standardisierung von Managementschnittstellen zu den vielfach bearbeiteten, aber größtenteils noch ungelösten Aufgaben im Systemmanagement. Viele Hersteller von Managementsystemen behaupten zwar, hier eine umfassende Lösung anbieten zu können, diese sind aber bei genauerer Betrachtung — insbesondere wegen der fehlenden Standardisierung auf diesem Gebiet — stets unvollständig.

In unseren bisherigen Arbeiten (z.B. [5]) haben wir das Problem durch einen Top-Down-Ansatz entlang typischer Überwachungs- und Managementaufgaben eines UNIX-Systemadministrators angegangen. Im Rahmen einer Untersuchung seiner Aufgaben wurde eine Systemmanagement-MIB entwickelt und implementiert, die (unter anderem) das Management folgender Objekte eines UNIX-Systems erlaubt:

- Speicher (Hauptspeicher, Swap-Datei)
- Geräte (Prozessoren, Drucker, Platten und Dateisysteme etc.)
- Prozesse
- Benutzer (Kenndaten, Aktivität, Gruppen, Quotas etc.)

Beim Design der daraus entwickelten SNMP-MIB (s. Abb. 3) wurde insbesondere Wert darauf gelegt, daß ein aktives, d.h. steuerndes Management — soweit sinnvoll — möglich ist[3].

[3] Die hierbei aufgeworfenen Sicherheitsprobleme (z.B. Änderung von Passworten) sind im Rahmen einer Diskussion über die Eignung von SNMP als Protokoll für das Management von Endsystemen zu diskutieren und nicht Gegenstand dieser Betrachtung.

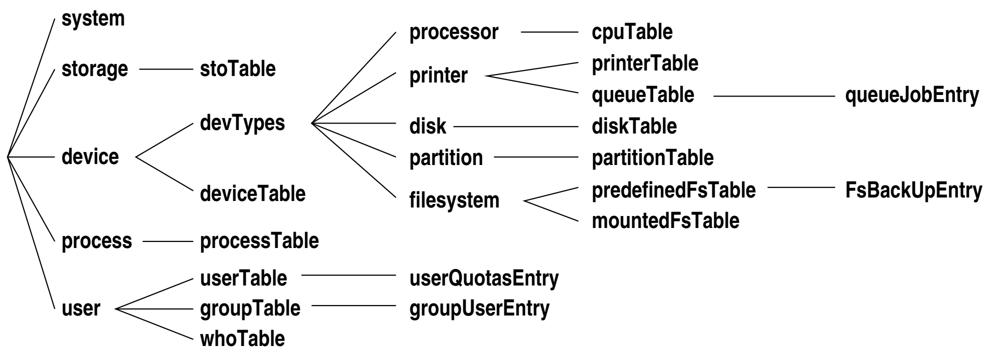


**Abbildung3.** Aufbau der Systemmanagement-MIB (nach [5])

Bei der Implementierung dieser MIB auf verschiedenen Betriebssystemen (SunOS, AIX, HP-UX) tritt die durch die einheitliche MIB verschattete Heterogenität der UNIX-Systeme natürlich wieder zu Tage. Hierbei hat es sich gezeigt, daß die bereits erwähnte, kaum vorhandene Standardisierung von UNIX-Managementschnittstellen in Form von APIs oder Konfigurationsdateien großen Aufwand bei der Portierung und Wartung der Agentensoftware erfordert.

Es mag eingewendet werden, daß der hierfür zu erbringende Aufwand in ähnlicher Größenordnung ist zu dem, der bei der Entwicklung von Managementlösungen entsteht, die z.B. auf der Ausführung von Skripten bzw. Befehlsfolgen via RSH, Perl, Tcl/TK oder RPC entstehen. Dabei wird jedoch übersehen, daß durch die Abstützung auf ein standardisiertes Managementprotokoll und durch die Verwendung von einheitlicher Managementinformation nur noch genau eine Managementarchitektur (anstelle vieler verschiedener Managersysteme) unterstützt werden muß.

Die von vielen Herstellern bereitgestellten, proprietären Managersysteme mit graphischer Benutzeroberfläche sehen zwar gut aus, sind aber für ein integriertes Management unterschiedlichster Endsysteme kaum hilfreich. Die Systemmanagement-MIB bietet hier für die angeführten Bereiche Abhilfe und fügt sich aufgrund der DPI-basierten Struktur nahtlos in die sonstige Agentensoftware des Endsystems ein.

### 3.2 Management von DHCP-Servern

Die zunehmende Dynamik in der Konfiguration von Netzen und die wachsende Anzahl an heterogenen Endsystemen (insbesondere PCs und mobile Endsysteme) führen dazu, daß Systemkonfiguration durch statische, dezentral (d.h. auf den Systemen selbst) und manuell zu pflegende Konfigurationsdateien immer weniger praktibel ist.

Die IETF Working Group *Dynamic Host Configuration* hat dieses Problem bereits vor einigen Jahren erkannt und als ersten Schritt hin zu einer mehr automatisierten und zentral gesteuerten Konfiguration das *Dynamic Host Configuration Protocol (DHCP)* entwickelt ([3, 14]). Derzeit finden intensive Bestrebungen statt, das Protokoll zu erweitern und für die Verwendung mit IPv6 und dem dann auch nötigen "Renumbering" anzupassen ([1]).

Das DHCP impliziert eine Client/Server-Struktur, in der die Clients (beispielsweise) zum Bootzeitpunkt Konfigurationsinformation vom DHCP-Server erfragen. Das hierbei abgewickelte Zwei-Phasen-Protokoll ist in Abb. 4 dargestellt. Die in der ersten Protokollphase (*DHCPDISCOVER*/*DHCPOFFER*) mögliche dynamische Auswahl eines

DHCP-Servers durch den Client erlaubt den Einsatz replizierter DHCP-Server und garantiert somit die Skalierbarkeit einer DHCP-Infrastruktur auch für Systemumgebungen mit einer großen Anzahl von Endsystemen.

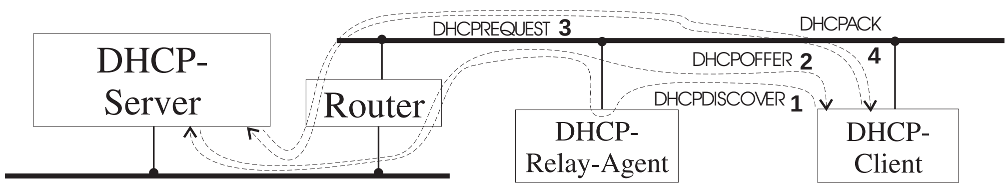


**Abbildung4.** Prinzipielle Architektur eines mittels DHCP konfigurierten Netzes

Die für das DHCP normierte Information umfaßt eine große, fortlaufend erweiterte Anzahl von Parametern z.B. zur Konfiguration von Netzinterfaces (IP-Adresse, Gateway, DNS-Server) und zu klassischen Basisanwendungen (Email-Server, Drucker-Queues etc.)

Der Einsatz von DHCP in stark dynamischen, zunehmend durch mobile Rechensysteme und PCs geprägten Umgebungen erscheint aus Effizienzgesichtspunkten für das Management als zwingend notwendig [7] und wird von den meisten gängigen Betriebssystemen auch bereits unterstützt. Es fehlt jedoch noch eine standardisierte Managementschnittstelle, die es ermöglichen würde, DHCP-Server unterschiedlicher Hersteller von einer Managementanwendung aus zu überwachen und zu steuern. Wir haben hierzu die in Abb. 5 angedeutete DHCP-MIB für DHCP-Server entwickelt und erweitern den Quellcode eines frei erhältlichen DHCP-Server-Dämonen um eine DPI-Schnittstelle. Damit ist aktives Management eines DHCP-Servers via SNMP möglich.

Die Tabelle *dhcpConfigTable* legt die Information fest, die der DHCP-Server auf Anfragen eines Clients hin übermittelt. Durch die Tabelle *dhcpLeaseTable* kann das Managementsystem Information über die derzeit aktiven Clients vom DHCP-Server erfragen, während durch den (unbestätigten) SNMP-Trap *dhcpLeaseStatusChange* Änderungen der Netzkonfiguration (wie z.B. das Auftreten neuer Clients) vom DHCP-Server an das Managementsystem gemeldet werden.

### 3.3 Die DQS-MIB

Moderne Arbeitsplatzrechner bieten eine hohe Rechenleistung, sind aber i.d.R. nur selten wirklich ausgelastet. Verteilte Batch-Systeme (oft auch als Distributed Queuing Systems (DQSs) oder Job Management Systems bezeichnet), die es erlauben, ungenutzte Ressourcen dieser Rechner z.B. für rechenzeitintensive Anwendungen zu verwenden, werden deshalb immer wichtiger (siehe z.B. [8]). Sie spielen, wenn sie intensiv genutzt werden, eine zentrale Rolle innerhalb der DV-Infrastruktur eines Unternehmens.

Das Management dieser Systeme basiert heute auf isolierten, spezialisierten Werkzeugen, die nicht oder nur rudimentär an andere Managementsysteme angebunden sind. Da aber z.B. bei der Konfiguration und Leistungsüberwachung offensichtliche Interdependenzen mit anderen Werkzeugen bestehen, sollten sie möglichst in ein integriertes Management der gesamten Infrastruktur eingebunden werden. Ein möglicher Weg besteht darin, eine „DQS-MIB“ zu entwickeln, die Management der DQSs auf der Basis eines standardisierten Managementprotokolls (hier SNMP) und der DPI-Schnittstelle erlaubt (siehe auch [9]). Die DQS-MIB stellt damit ein weiteres Beispiel

```
dhcpConfigTable OBJECT-TYPE                    -- Table of configuration
    SYNTAX      SEQUENCE OF DhcpConfigEntry .. -- entries for the DHCP server

DhcpConfigEntry ::= SEQUENCE {
    dhcpConfigName          DisplayString,
    dhcpConfigParent        DisplayString,  -- for inheritance from
    dhcpConfigIpAddrFirst   IpAdress,       --      dhcpConfigName
    dhcpConfigIpAddrLast    IpAdress,
    dhcpConfigPhysAddress   OCTET STRING,
    dhcpConfigPhysType      INTEGER,
    dhcpConfigGateway       IpAddress,
    dhcpConfigSubnetMask    IpAddress,
    dhcpConfigNameServer    IpAddress,
    dhcpConfigMobileIpAgent DisplayString, ... }

dhcpLeaseTable  OBJECT-TYPE    -- Table of leases handed out by the DHCP server
    SYNTAX      SEQUENCE OF DhcpLeaseEntry ...

DhcpLeaseEntry ::= SEQUENCE {
    dhcpLeaseIpAddr         IpAdress,     dhcpLeaseBegin        DateAndTime,
    dhcpLeasePhysAddress    OCTET STRING, dhcpLeaseDuration     TimeTicks,
    dhcpLeasePhysType       INTEGER,      dhcpLeaseTimeLeft     TimeTicks,
                                          dhcpLeaseStatus       INTEGER, ... }

dhcpLeaseStatusChange   TRAP-TYPE         -- Trap for changes in lease status
    ENTERPRISE dhcp
    VARIABLES { dhcpLeaseIpAddr, dhcpLeasePhysAddress, dhcpLeaseBegin,
                dhcpLeaseStatus, dhcpLeasePhysType, dhcpLeaseTimeLeft}
```

**Abbildung5.** Auszug eines Vorschlags für eine DHCP-MIB

für ein Ressourcen-Modul zu einer verteilten Anwendung dar, das in einen modularen Agenten einzubinden ist.

## 4 Implementierung multiarchitektureller Agenten

Während in Abschnitt 3 die Entwicklung von Agenten für heterogene Systeme im Rahmen einer bestimmten Managementarchitektur beschrieben wird, befaßt sich dieser Abschnitt mit der Implementierung von Agenten, die *mehrere Managementarchitekturen zugleich* unterstützen. Im Blickpunkt der Untersuchung stehen die Internet–Managementarchitektur sowie die *Common Object Request Broker Architecture (CORBA)*, die Bestandteil der *Object Management Architecture (OMA)* ist.

Da einerseits SNMP–fähige Agenten weit verbreitet und andererseits CORBA–Agenten noch kaum vorhanden sind, liegt es nahe, Verfahren für die Erweiterung bestehender SNMP–Agentenimplementierungen zu entwickeln, damit auf die in ihnen enthaltene Managementfunktionalität auch via CORBA zugegriffen werden kann. Aufgrund der hohen Flexibilität bezüglich der verwendeten Managementarchitektur, die das Konzept des „Multiarchitekturellen Agenten“ (siehe Abschnitt 1) bietet, haben wir uns für unsere Agentenimplementierung, die auf der in Abschnitt 3.1 beschriebenen

Systemmanagement-MIB aufbaut, entschieden, die notwendigen Abbildungen zwischen den Architekturen bereits im Agenten durchzuführen.

Geht man von vorhandenen SNMP–Agenten aus, können CORBA–Agenten in zwei Schritten entwickelt werden:

– Zuerst müssen die Beschreibungen der SNMP–Managementobjekte in das CORBA–Objektmodell überführt werden; es handelt sich hierbei um eine geeignete Abbildung der Managementobjekt–Schnittstellen, die zum Teil maschinell erfolgen kann (siehe Abschnitt 4.1).
– Anschließend wird die Managementfunktionalität des SNMP–Agenten auf das CORBA–System portiert, indem man vorhandenen Agentencode übernimmt und als CORBA-konforme Objekte kapselt (siehe Abschnitt 4.2).

## 4.1 Transformation von SNMP–MIBs in CORBA–Objektbeschreibungen

Wichtigster Gesichtspunkt bei der Überführung von Agenten einer Managementarchitektur in eine andere Architektur ist die unterschiedliche Mächtigkeit der jeweiligen Informationsmodelle. Während das Internet-Informationsmodell sämtliche Aspekte eines Agenten (Parameter und Aktionen) in Form von skalaren Datentypen bzw. Tabellen beschreibt und – im Gegensatz zu CORBA – keine Mechanismen wie Vererbung und Allomorphie kennt, werden im CORBA-Objektmodell Agenten in Form von Objektklassen sowie den dazugehörigen Attributen und Methoden definiert. Die Konsequenzen dieser Unterschiede sind nachfolgend skizziert.

Die Abbildung bestehender Objektbeschreibungen in das CORBA–Objektmodell bedingt die algorithmische Überführung der Spezifikationssprachen, in denen die Managementobjekte beschrieben sind. In unserem Fall handelt es sich um eine Übersetzung der in der Internet–Managementarchitektur verwendeten ASN.1–Templatesprache in die OMG *Interface Definition Language (IDL)*. Für diesen Zweck existiert ein Algorithmus [15], der sich zur Zeit in der Standardisierungsphase befindet und die Transformation der in SNMPv2 vorhandenen Datentypen, Makros und asynchronen Ereignismeldungen in die CORBA–Entsprechungen vornimmt:

Für jede Gruppe einer SNMP–MIB wird eine Objektklasse erzeugt; die darin enthaltenen einfachen skalaren Datentypen werden zu Attributen der Objektklasse. So werden zum Beispiel `Integer32` bzw. `TimeTicks` auf die IDL–Datentypen `long` bzw. `unsigned long` abgebildet; `DisplayString` und `IpAddress` werden einer Sequenz von Octets zugeordnet.

SNMP–Tabellen werden durch den Algorithmus ebenfalls zu Objektklassen transformiert: Jede Zeile einer Tabelle wird damit zu einer Instanz einer Objektklasse, die durch eine IDL–Schnittstelle festgelegt ist. Ein Beispiel soll dies erläutern: Enthält ein System drei Festplatten, so wird dies auf der SNMP–Seite durch das Anlegen von drei Zeilen der Tabelle `stoTable` dargestellt. CORBA–seitig würden stattdessen drei Instanzen der Objektklasse `StorageDevice` erzeugt. Das CORBA-Objektmodell ist hierbei dem intuitiven Verständnis näher als das entsprechende Internet-Informationsmodell.

Variablen, die einzelne Felder der Tabelle spezifizieren, werden im Falle einfacher Datentypen zu Attributen der Objektklasse.

Obwohl mit dem Übersetzungsalgorithmus dem Entwickler ein mächtiges Werkzeug zur syntaktischen Überführung von SNMP–Objektbeschreibungen in das CORBA–Objektmodell zur Verfügung steht, sind manuelle Eingriffe erforderlich, da Managementsemantik im Internet-Informationsmodell oft nur implizit definiert ist: So wird das Auslösen von Aktionen auf Managementobjekten, für das in der Internet-Managementarchitektur kein explizites Sprachmittel vorgesehen ist, durch das Setzen entsprechender MIB–Variablen (sogenannter *Pushbutton*–Variablen) simuliert. Das CORBA–Analogon hierzu besteht im Aufruf einer Methode auf dem jeweiligen Managementobjekt. Der Übersetzungsalgorithmus, der die Abbildung der SNMP–Datentypen in äquivalente IDL–Konstrukte vornimmt, müßte also eine Pushbutton–Variable auf eine Methode des Managementobjektes abbilden. Da Pushbutton–Variablen jedoch nicht syntaktisch erkennbar sind, muß diese Transformation manuell durch den Entwickler erfolgen. Eine MIB-Variable `stoFormat`, deren Setzen die Formatierung einer Festplatte veranlaßt, muß daher manuell auf eine Methode `storage_format` der Objektklasse `Storage` abgebildet werden, da sie ansonsten maschinell zu einem einfachen Attribut würde. Auch in diesem Fall zeigt sich, daß das CORBA-Objektmodell dem betrachteten Problembereich angemessener ist als die Internet-Variante.

### 4.2 Überführung bereits existierender Managementfunktionalität

Nachdem die Beschreibung der Objektklassen mit ihren Attributen und Methoden erzeugt wurde, muß nun in einem zweiten Schritt die Umsetzung der Funktionalität erfolgen, die die Nutzung der Managementinformation erst ermöglicht. Der zu erweiternde Agent soll sowohl weiterhin unter SNMP genutzt werden als auch von einem CORBA-konformen Managementsystem aus zugreifbar sein.

Dies ist gleichbedeutend mit der in zahlreichen Bereichen der Informatik auftretenden Fragestellung, wie bestehende Altsysteme (*legacy systems*) schonend in die objektorientierte Welt migriert werden können. Zentraler Gedanke ist hierbei, unter Zuhilfenahme sogenannter *Wrapper* bestehenden Code geeignet in Objektklassen zu kapseln und somit für neue objektorientierte Systeme zugreifbar zu machen. Der betrachtete SNMP-Agent ist in der Programmiersprache C implementiert worden und genügt damit keinesfalls objektorientierten Prinzipien. Dies ist jedoch für die Migration ohne Bedeutung, da für objektorientierte Applikationen der Begriff des „Perception is reality“ gilt. Es ist also nicht erforderlich, daß ein System objektorientiert implementiert worden ist; maßgeblich ist jedoch, daß seine Bestandteile wie Objekte aussehen.

Sehr wichtige Voraussetzungen für eine Kapselung in hinreichend feiner Granularität sind, daß das zu migrierende System einerseits genügend modular implementiert worden ist und andererseits die Schnittstellen seiner Prozeduren offengelegt sind bzw. die Prozeduren im Quellcode vorliegen. Beides ist in unserem Fall gegeben: Jede MIB-Variable ist in einem eigenen Modul implementiert worden und verfügt über eine bzw. zwei Schnittstellen je nachdem, ob auf die Variable nur lesend oder auch schreibend zugegriffen werden kann. Die ausschließlich lesbare MIB-Variable `sysName` wird durch Aufruf der Prozedur get_sysName ermittelt; eine schreib- und lesbare Variable wie `sysLocation` wird durch die Prozeduren get_sysLocation und set_sysLocation implementiert.

Wir sind daher folgendermaßen vorgegangen: Die IDL–Schnittstellenbeschreibungen, die durch die in Teilabschnitt 4.1 erläuterten Umsetzungen erhalten wurden, werden nun einem IDL–Compiler übergeben, der einerseits die Schnittstellen der neu erstellten Objektklassen (hier: die *Wrapper* für die bereits bestehenden Prozeduren) innerhalb des ORB–Laufzeitsystems bekanntmacht und andererseits die Datenstrukturen und Schnittstellen des CORBA–Agenten in einer herkömmlichen Programmiersprache (im betrachteten Fall: C) generiert.

Die so entstandenen C-Programmrümpfe des CORBA–Agenten können nun um die entsprechenden Aufrufe von Prozeduren des bestehenden SNMP–Agenten ergänzt werden, deren Aufgabe lediglich darin besteht, die erhaltenen Parameter auf die Parameter der bestehenden Agentenprozeduren abzubilden und anschließend diese Prozeduren aufzurufen. Hierbei ist es von großem Vorteil, daß der Agent modular aufgebaut ist und die Schnittstellen des Agenten sowohl zu den Ressourcen als auch zum Managementprotokoll hin klar definiert sind. Nach Abschluß dieser Arbeiten ist der bestehende SNMP–Agent um eine CORBA–Schnittstelle erweitert; der entstandene multiarchitekturelle Agent kann nun sowohl unter SNMP als auch unter CORBA genutzt werden.

In den von uns durchgeführten Arbeiten konnte die Erfahrung gewonnen werden, daß sich der Aufwand für die Kapselung des bestehenden Programmcodes durch die neu entwickelten Schnittstellen dank der modularen Implementierung des SNMP-Agenten in sehr engen Grenzen hält: Die vollständige Kapselung des Agenten, dessen MIB aus insgesamt 165 Variablen und 15 Tabellen besteht, wurde innerhalb einer Mannwoche durchgeführt.

Zur Performance des Systems ist folgendes anzumerken: Da der Zugriff über die SNMP-Schnittstelle unverändert geblieben ist, ergibt sich unter dem SNMP-Gesichtspunkt verständlicherweise keinerlei Veränderung zum vorherigen Stand; CORBA-seitig sind insbesondere beim Transfer größerer Datenmengen, wie sie z.B. beim Auslesen der Prozeßtabelle anfallen, signifikante architekturbedingte Performancevorteile ersichtlich. Diese werden jedoch durch Unzulänglichkeiten der momentan verfügbaren CORBA-Implementierungen kompensiert, im Falle einfacher skalarer Variablen sogar (leider) überkompensiert. Die architekturbedingten Performancevorteile liegen darin begründet, daß das unter SNMP sehr ineffiziente Durchlaufen von Tabellen mit dem `get-next`-Operator entfällt. Die Unzulänglichkeiten der CORBA-Toolkits beruhen auf der Implementierung des *Interface Repository* in Form einfacher Dateien, deren Verzeichnisse von den Maschinen, auf denen der Object Request Broker läuft, wechselseitig über das *Network File System* exportiert bzw. gemountet werden müssen.

Es handelt sich hierbei jedoch um „Kinderkrankheiten“, mit denen neue Systeme zwangsläufig behaftet sind und an deren Behebung die Hersteller arbeiten; zukünftige CORBA-Implementierungen werden andere Verfahren nutzen, um den Zugriff auf verteilte Daten effizient handhaben zu können.

Insgesamt konnte die prinzipielle Eignung von CORBA für die Belange des Systemmanagements nachgewiesen werden, da das zugrundeliegende Objektmodell die Möglichkeit bietet, Klassen von Managementobjekten sowie deren Attribute und Methoden eleganter und intuitiver zu implementieren, als dies unter der Internet-Managementarchitektur möglich ist. Die Migration vorhandener, modular aufgebauter SNMP-Agenten ist unproblematisch, sofern die Schnittstellen der Agentenprozeduren offengelegt sind.

## 5 Zusammenfassung und Ausblick

Der Beitrag hat gezeigt, wie mit der Hilfe offener, standardisierter Schnittstellen modulare Managementagenten realisiert werden können, deren Module herstellerübergreifend kombinierbar sind. Die Agenten sind damit zur Laufzeit flexibel konfigurierbar, sie können prinzipiell an unterschiedlichste Einsatzbedingungen, Umgebungen und Betreiberanforderungen angepaßt werden. Damit ist eine wichtige Vorausetzung für integriertes Management in offener, heterogener Umgebung gegeben.

Derzeit existieren allerdings mehrere konkurrierende Managementarchitekturen und als Folge davon auch unterschiedliche Ansätze für die genannte Schnittstelle. Dies führt häufig zu einer abwartenden Haltung der SW-Hersteller und zu einer Verzögerung bei der Einführung des von den Betreibern dringend benötigten integrierten Managements. Da unwahrscheinlich ist, daß sich genau eine dieser Architekturen durchsetzen wird, müssen möglichst schnell Übergänge geschaffen werden. Der Beitrag hat einen ersten Ansatz dazu im Hinblick auf die Agentenimplementierung skizziert.

Sind die genannten Architekturprobleme gelöst, bleiben trotzdem noch diverse Fragen im Bereich des integrierten Managements offen. Die wichtigste besteht in der Definition der Managementinformation, also der herstellerunabhängigen Modellierung der zu administrierenden Ressourcen. Speziell im Bereich der verteilten Anwendungen und der Endsysteme fehlen hier noch Lösungen, die dann die Basis für die Entwicklung der zugehörigen Ressourcen-Module legen. Weitere offene Fragen sind zu klären, wenn man vom derzeit verbreiteten passiven Monitoring zu (pro-)aktivem integriertem Management kommen will.

*Danksagung*
Die Autoren danken dem Münchner Netzmanagement Team für intensive Diskussionen zu früheren Versionen dieses Beitrags. Das MNM-Team ist eine Gruppe von Wissenschaftlern beider Münchner Universitäten und des Leibniz Rechenzentrums der Bayerischen Akademie der Wissenschaften.

## References

1. Bradner, S. O., Mankin, A.: IPng - Internet Protocol Next Generation. IPng Series. Addison-Wesley 1995
2. Desktop Management Interface Specification. Version 1.0. Desktop Management Task Force. (April 1994)
3. Droms, R.: RFC 1541: Dynamic Host Configuration Protocol. Technical Report. IAB. (Oktober 1993)
4. Freed, N., Kille, S.: RFC 1565: Network Services Monitoring MIB. Technical Report. IAB. (Januar 1994)
5. Gutschmidt, M., Neumair, B.: Integration von Netz- und Systemmanagement: Ziele und erste Erfahrungen. In: *Proceedings der 3. Fachtagung Arbeitsplatzrechensysteme (APS'95), Hannover.* Mai 1995
6. Hegering, H.-G., Abeck, S.: Integrated Network and System Management. Addison-Wesley 1994
7. Heilbronner, S., Neumair, B.: Management mobiler Endsysteme: Anforderungen bei der Integration in bestehende Managementsysteme. In Wall, D. (Hrsg.): *Organisation und Betrieb von DV-Versorgungsstrukturen.* Göttingen, Germany: Deutscher Universitäts-Verlag, November 1995, pp.117–131. ISBN 3-8244-2069-4
8. Kaplan, J. A., Nelson, M. L.: A Comparison of Queueing, Cluster and Distributed Computing Systems. Technical Report. NASA Langley Research Center. (Juni 1994)
9. Neumair, B., Wies, R.: Applying Management Policies to Manage Distributed Queuing Systems. Distributed Systems Engineering Journal *3*(3), 96–103 (1996)
10. Rose, M. T.: The Simple Book — An Introduction to Internet Management. Zweite Aufl. Prentice-Hall 1994
11. Sloman, M. S. (Hrsg.): Network and Distributed Systems Management. Addison Wesley 1994
12. Sturm, R., Weinstock, J.: Application MIBs: Taming the Software Beast. Data Communications, (November 1995)
13. Wijnen, B., Carpenter, G., Curran, K., Sehgal, A., Waters, G.: Simple Network Management Protocol Distributed Protocol Interface Version 2.0. RFC 1592. IAB. (März 1994)
14. Wobus, J.: DHCP FAQ. USENET Frequently Asked Questions. Dezember 1995. http://web.syr.edu/ jmwobus/comfaqs/dhcp.faq.html
15. Inter-Domain Management Specifications: Specification Translation (Sanity Check Draft). Preliminary Specification Pxxx. X/Open Ltd. (August 1995)