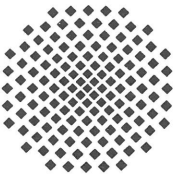


Universität Stuttgart

Institut für Nachrichtenvermittlung und Datenverarbeitung
Prof. Dr.-Ing. habil. Dr. h.c. P. J. Kühn

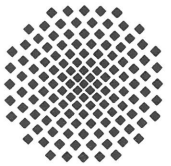
65. Bericht über verkehrstheoretische Arbeiten

Überwachung von Rechnernetzen mit verteilten Meßsystemen

von
Werner Schollenberger
1996

D 93

Druck: E. Kurz & Co., Druckerei + Reprografie GmbH., Stuttgart
ISBN 3-922403-75-1



University of Stuttgart

Institute of Communication Networks and Computer Engineering
Prof. Dr.-Ing. habil. Dr. h.c. P. J. Kühn

65th Report on Studies in Congestion Theory

Monitoring Computer Networks Using Distributed Measurement Systems

by
Werner Schollenberger

1996

Monitoring Computer Networks Using Distributed Measurement Systems

Summary

With the growth of the number of business processes which rely on computer networks, the availability and functionality of these networks is becoming a more important issue. Hence, continuous network monitoring plays an essential role in network management. Being able to quickly identify problems and partial breakdowns and to locate their origin is one of the main goals of network operation. This report deals with the problem of deploying measurement systems for the effective and efficient monitoring of networks. It investigates how a network administrator can be supported in the job of monitoring a network.

Chapter 1 introduces computer networks and network management and positions the latter within the OSI seven layer reference model. The major differences between the management of Local Area Networks and Public Networks are identified.

Chapter 2 gives an overview of network monitoring. It starts with a detailed introduction to network management standards, followed by a classification of existing measurement systems. Then, the measurement principles used in communication networks are described. Further, standardized approaches to modelling information recorded by measurement systems are presented together with the monitoring functions found in OSI network management standards. Finally, the monitoring capabilities of current management tools and platforms are outlined.

Network management standards concentrate on the infrastructure required to centrally control network components using network management protocols. They cover the modelling and exchange of management information. However, the way these standards should be applied in order to actually manage a network is not addressed. On the other hand, a number of measurement systems are available. Being distributed throughout the network, they are able to perform monitoring tasks. Measurement systems differ widely in their functionality and in the way they are remotely controlled. Only a few of them support standardized management protocols, with the others using proprietary mechanisms.

Chapter 3 presents a new concept which integrates the wide variety of measurement systems and provides the network administrator with methods to define and organize the monitoring activities. This concept assumes that the administrator knows which information about the network is of interest and has to be collected. A monitoring system gathers the raw data from

the various distributed measurement systems, selects the relevant information and processes it according to the administrator's selection criteria.

A monitoring activity can itself be treated as a measurement. In this model, the distributed measurement systems act as sensors and are analogous to measurement instruments or probes. The measurement itself filters the data recorded by the sensors and calculates one or more meaningful measurement results. A measurement may consist of periodically recurring sub-measurements. Since the monitoring objectives of computer networks cannot be defined in general, it is important that the user is able to parameterise measurements. Timing, measurement system selection, measurement specific information and measurement presentation can all be parameterised. In order to improve performance the network topology and measurement configuration and their interdependencies are stored in a relational database.

Based on this concept, a prototype Monitoring System has been designed and implemented. It enables the user to create monitoring activities using a high level language, and automatically initialises and controls them. It provides for the management of a variety of measurements and for communication with distributed measurement systems. From the data received the measurement results are calculated and presented to the user.

Chapter 4 is dedicated to the Monitoring System's architecture which is had been designed as a concurrent multi-process system. For the sake to system stability and expandability, the functions of measurement creation, execution, and result processing are divided into three co-operating processes. The interprocess communication mechanisms allow new measurements and measurement equipment to be easily inserted and ensure high performance. A reservation scheme for measurement systems guarantees that collisions between different running measurements trying to accessing a single component will be resolved within an acceptable time frame. Limited testing of the prototype took place in the faculty network.

Chapter 5 investigates how the Monitoring System would behave under certain load conditions in a real production environment. In order to avoid the high costs of performing experiments using a prototype in a testbed configuration, the system performance is evaluated using Event Simulation. The model of the Monitoring System takes into account the system architecture with its processes, synchronisation points, and process interaction mechanisms, in their relevance to system performance. This leads to a rather complex model with several levels of hierarchy. An object oriented simulation approach succeeded in handling this complexity. Based on processing times derived from measurements with a prototype implementation performance values for particular types of measurements are presented. Conclusions are drawn about the loading the system can tolerate before appreciable processing delays become apparent.

Inhaltsverzeichnis

Abkürzungen

ACSE	Association Control Service Element
API	Application Programming Interface
AEZ	Auftragserzeugung
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
BMA	Basismeßauftrag
BTX	Bildschirmtext
CIM	Computer Integrated Manufacturing
CMIP	Common Management Information Protocol
CMISE	Common Management Information Service Element
CM-PDU	Common Management Protocol Data Unit
CPU	Central Processing Unit
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DME	Distributed Management Environment
DQDB	Distributed Queue Dual Bus
EAB	Ergebnisaufbereitung
EFD	Event Forwarding Descriptor
ER	Entity-Relation
EWMA	Exponentially Weighted Moving Average
FDDI	Fiber Distributed Data Interface
FIFO	First In, First Out
GDMO	Guidelines for the Definition of Management Objects
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMC	Inter Module Communication
IN	Intelligent Network (Intelligentes Netz)
IO	Input-Output
IP	Internet Protocol
IPC	Inter Process Communication
ISDN	Integrated Services Digital Network

ISO	International Organization for Standardization
KI	Künstliche Intelligenz
LAN	Local Area Network
LLC	Logical Link Control
LLTP	Low Level Transport Protocol
MA-PDU	Management Application Protocol Data Unit
MAC	Media Access Control
MF	Mediation Function
MIB	Management Information Base
MOT	Managed Object Under Test
MTS	Monitoring-System
NEF	Network Element Function
OSI	Open Systems Interconnection
OSF	Operations System Function
PEARL	Process and Experiment Automation Real-time Language
PDU	Protocol Data Unit
QAF	Q-Adaptor Function
QOS	Quality of Service
RDN	Relative Distinguished Name
RFC	Request for Comment
RMON	Remote Monitoring
ROSE	Remote Operations Service Element
SCO	Santa Cruz Operation Inc.
SDU	Service Data Unit
SHM	Shared Memory
SLIP	Serial Line Interface Protocol
SMAE	Systems Management Application Entity
SMASE	Systems Management Application Service Element
SMF	Systems Management Function
SMFA	Systems Management Functional Areas
SNMP	Simple Network Management Protocol
SNMPv2	Version 2 of the Simple Network Management Protocol
SQL	Structured Query Language
TAP	Transceiver Access Point
TARR	Test Action Request Receiver
TCL	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TMN	Telecommunications Management Network
TO	Test Object
UDP	User Datagram Protocol

UIF	User Interface
UWMA	Uniformly Weighted Moving Average
WMF	Workload Monitoring Function
WSF	Workstation Function
WWW	World Wide Web
ZDS	Zwischendatenstruktur

Kapitel 1

Einleitung

Wir leben in einem Zeitalter der Informationen und des Informationsaustausches, in dem untereinander vernetzte Rechnersysteme eine immer größer werdende Rolle spielen. In vielen Bereichen des wirtschaftlichen Lebens herrscht mittlerweile eine hochgradige Abhängigkeit von den Rechnernetzen, sei es im Bereich der computerintegrierten Fertigung (Computer Integrated Manufacturing, CIM) oder der Büroautomatisierung. Immer mehr erfolgt der schnelle Zugriff auf zentrale Datenbestände – wie etwa bei Buchungsvorgängen, beim Zugriff auf Kundendaten oder im Zahlungsverkehr – rechnerunterstützt, und immer mehr kommt auch der sogenannte Normalbürger damit in Kontakt. Die Abwicklung von Bestellungen oder Bankgeschäften über Bildschirmtext (BTX) oder der massive Anstieg der Anzahl von Anbietern eines Zugangs zu dem weltumspannenden *Internet* [26], [86] bzw. dem *World Wide Web* (WWW) [101] sind hier nur willkürlich herausgegriffene Beispiele.

Gleichzeitig erfuhr die Rechnerkommunikation in den letzten Jahren einen starken Strukturwandel. Ursprünglich war sie darauf ausgerichtet, intelligente Peripheriegeräte an einen zentralen Großrechner zu koppeln. Die Fortschritte in der Halbleitertechnologie und die Erkenntnis, daß viele kleinere Rechnersysteme preisgünstiger herzustellen sind als ein System, dessen Leistung der Summe der Einzelsysteme entspricht, führte zu einer Entwicklung, die unter dem Namen *Downsizing* bekannt wurde. Dies führt zu einer Ablösung der Mainframes durch eine Reihe von untereinander vernetzten Workstations, die einzelnen Benutzern zugeordnet sind. Dadurch entstehen dezentrale Rechnerumgebungen, in denen ein hoher Kommunikationsbedarf herrscht, um den Informationsaustausch zwischen den Anwendern, einen ortsunabhängigen Zugriff auf Daten und Betriebsmittel sowie die verteilte Ausführung großer Applikationen zu realisieren. Die Vernetzung schafft eine so große Abhängigkeit zwischen den Systemen, daß diese beim Ausfall der Kommunikationsinfrastruktur oft nicht mehr oder nur sehr beschränkt arbeitsfähig sind. Deshalb ist das Netzmanagement zu einer der wichtigsten und dringlichsten Aufgaben im Bereich der Kommunikationstechnik avanciert.

1.1 Rechnernetze und deren Management

Rechnernetze erlauben den daran angeschlossenen Systemen miteinander zu kommunizieren. Damit die Kommunikation erfolgreich ist, müssen jedoch eine Menge von Regeln eingehalten werden. Die International Organization for Standardization (ISO) verabschiedete mit dem *Basisreferenzmodell zur Verbindung offener Systeme* (Open Systems Interconnection, OSI) [48] ein wichtiges Hilfsmittel zur systematischen Gliederung der vielfältigen Kommunikationsaufgaben und einen Rahmen, in dem die verschiedenen Kommunikationsaspekte standardisiert werden konnten. Wie in Bild 1.1 dargestellt, ist das Modell in sieben Schichten unterteilt, die jeweils bestimmte Aspekte der Kommunikation abdecken.

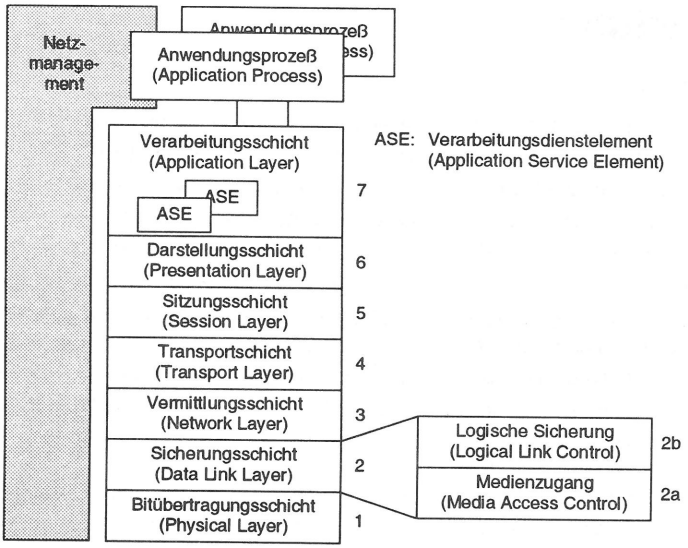


Bild 1.1: Das OSI-Referenzmodell für die Kommunikation offener Systeme

Jede Schicht realisiert einen bestimmten Dienst, den sie der übergeordneten Schicht anbietet. Für die Erbringung dieses Dienstes werden *Protokolldateneinheiten* (Protocol Data Units, PDUs) mit einer Instanz derselben Schicht des Kommunikationspartners ausgetauscht. Für den Austausch dieser PDUs wird wiederum der Dienst der darunterliegenden Schicht in Anspruch genommen. Die Dienstschnittstelle wird über *Dienstprimitive* (Service Primitives) definiert, die wiederum jeweils eine *Dienstdateneinheit* (Service Data Unit, SDU) enthalten.

Die Dienste und Kommunikationsprotokolle der verschiedenen Schichten sind in einzelnen Standards festgeschrieben. Für einige der Schichten existieren mehrere Standards, die unterschiedliche Varianten des Dienstes bzw. des Protokolls definieren. Abhängig vom Einsatzge-

biet werden zueinander passende Standards für jede Schicht kombiniert und bilden ein *Protokollprofil*. Während sich die Schichten 1 bis 6 allgemein mit der Übertragung von Daten befassen, ist die *Verarbeitungsschicht* auf die Kommunikationsbedürfnisse der *Anwendungsprozesse* zugeschnitten. Sie setzt sich aus *Verarbeitungsdienstelementen* (Application Service Elements, ASEs) zusammen, die den Anwendungsprozeß bei bestimmten Kommunikationsaufgaben unterstützen, wie z.B. dem Dateitransfer [50], der Steuerung von Fertigungseinrichtungen [59] oder auch dem Netzmanagement [60], [61].

Bei *lokalen Netzen* (Local Area Networks, LANs) ist die Schicht 2, wie in Bild 1.1 gezeigt, in zwei Teilschichten aufgespalten. Damit ist der Zugriffsmechanismus auf das Medium (Schicht 2a, Media Access Control, MAC) von der Funktionalität der gesicherten Übertragung einzelner Rahmen und der Adressierung der nächsthöheren Instanz (Schicht 2b, Logical Link Control, LLC) unabhängig standardisierbar. Die Schichten 1 und 2a sind abhängig von der Netztechnologie und in gemeinsamen Dokumenten des Institute of Electrical and Electronics Engineers (IEEE) und der ISO genormt [53]-[56], [58]. Die Schicht 2b und damit die Dienstschnittstelle zur Vermittlungsschicht ist für alle Netztechnologien gemeinsam.

Das starke Anwachsen der Größe von Netzinstallationen – mehrere hundert Stationen sind kein Sonderfall – und die Bedeutung der Netze im täglichen industriellen Betrieb führten dazu, daß das Netzmanagement eine zentrale Bedeutung erlangte und sich zu einem Hauptschwerpunkt der Forschung und Entwicklung im Bereich der Kommunikationstechnik entwickelte. Dies geht so weit, daß bei neuen Netztechnologien Managementaspekte bereits bei der Entwicklung mit berücksichtigt werden und in die Standards mit einfließen [1].

Betrachtet man reale Netze unter dem Gesichtspunkt des Netzmanagements, können sie in lokale Netze und öffentliche Netze eingeteilt werden. Lokale Netze ermöglichen die freie Datenkommunikation innerhalb einer Organisationseinheit. Unter dem Begriff Organisationseinheit kann eine Abteilung, ein Standort oder ein ganzes Unternehmen verstanden werden. Die Netze können in untereinander verbundene Teilnetze strukturiert und mit anderen lokalen Netzen über Weitverkehrsverbindungen gekoppelt sein. Innerhalb eines Teilnetzes wird das Übertragungsmedium von allen angeschlossenen Stationen gemeinsam benutzt (*Shared Medium*), wobei entsprechende Medienzugriffsprotokolle den verteilten Zugriff regeln und die zur Verfügung stehende Übertragungsbandbreite möglichst gerecht aufteilen. Dem Benutzer steht die volle Medienbandbreite zur Verfügung, die er sich jedoch mit anderen teilen muß, d.h. die Benutzer beeinflussen sich gegenseitig und können das Netz auch überlasten. Bis zur Schicht 3 erfolgt die Kommunikation verbindungslos über Datagramme. Die Netztopologien entsprechen traditionellerweise Bus- oder Ringstrukturen, werden aber bei Neuverkabelungen überwiegend in Sternform mit zentralem Knoten, der jedoch keine Vermittlungseigenschaften besitzt, realisiert. Das Management für lokale Netze zeichnet sich durch folgende Punkte aus:

- Das Netz besteht aus heterogenen Komponenten.
- Im Netz sind mehrere Kommunikationsprotokolle im Einsatz.
- Das Netzmanagement schließt die Endeinrichtungen mit ein.
- Die Verantwortung für das Netz liegt bei der Organisationseinheit, in der das Netz benutzt wird und von der es auch über ein zentrales Netzmanagement verwaltet wird.

Öffentliche Datennetze arbeiten im Gegensatz zu LANs grundsätzlich verbindungsorientiert, auch wenn für den Benutzer verbindungslose Dienste angeboten werden. Der Anwender sieht vom Netz lediglich den Netzzugang, an dem ein bestimmter Dienst mit definierten Güteparametern zur Verfügung steht. Bei öffentlichen Netzen ist der Vermittlungsaspekt wesentlich stärker ausgeprägt. Der Verkehr der einzelnen Teilnehmer wird stark gebündelt und auf hochbitratigen Datenpfaden über mehrere Vermittlungsstellen bis zum Kommunikationspartner geleitet. Die Übertragungskapazität des Netzes ist wesentlich größer als die eines einzelnen Anwenders, und die gegenseitige Beeinflussung der Benutzer wird durch den Betreiber auf ein Minimum reduziert. Das Netzmanagement unterscheidet sich wie folgt von dem lokaler Netze:

- Das Netz wird von einer Betreiberorganisation verwaltet, deren Zuständigkeit an der Netzzugangsschnittstelle endet.
- Es existieren interne, logische Netze für Signalisierinformationen, deren Management dem in LANs ähnelt.
- Im Nutzdatenbereich endet das Netzmanagement an der Schicht 3, d.h. dem Netzzugang.
- Öffentliche Netze sind zu groß für ein zentrales Netzmanagement.

Ein wichtiger Aspekt des Netzmanagements ist die Netzüberwachung. Sie erlaubt es, Störungen und Fehler frühzeitig zu erkennen und umfaßt Statistiken über die Benutzung des Netzes. In der Netzüberwachung wird in ganz erheblichem Maße Gebrauch von Meßsystemen gemacht, die im Netz verteilt installiert werden und Informationen über das Netz vor Ort erfassen.

1.2 Ziel dieser Arbeit

Diese Arbeit befaßt sich mit dem Problem, Meßsysteme für eine Netzüberwachung sinnvoll und effizient einzusetzen. Hierzu werden die standardisierten Netzmanagementansätze unter dem Blickwinkel der Überwachung beleuchtet. Die Standardisierungsaktivitäten im Bereich des Netzmanagements konzentrieren sich auf die Schaffung einer Infrastruktur, die es erlaubt, Netzkomponenten zentral unter Verwendung eines Netzmanagementprotokolls zu verwalten. Dabei stehen die Struktur und Modellierung der Managementinformationen sowie der Transport und die Verwaltung dieser Informationen im Mittelpunkt. Auf die Anwendung dieser Standards wird jedoch bewußt nicht eingegangen. Bei den für lokale Netze verfügbaren verteilten Meßsystemen variieren sowohl die Meßfunktionen als auch die Art der zentralen Steuerung

sehr stark, und die Unterstützung von standardisierten Netzmanagementprotokollen ist nur teilweise gegeben. Der Schwerpunkt der Arbeit liegt in der Erstellung eines Konzeptes zur Integration der verschiedenen Meßsysteme und der Entwicklung einer Systemarchitektur, die dieses Konzept umsetzt. Der Ansatz geht davon aus, daß der Netzadministrator letztendlich weiß, welche Größen in seinem Netz wie erfaßt werden sollen und daß er für die Durchführung der dafür erforderlichen Überwachungsvorgänge und den effizienten Einsatz der Meßsysteme eine Rechnerunterstützung benötigt. Zwar beschränken sich die Betrachtungen in dieser Arbeit auf lokale Netze, das Konzept sollte sich jedoch auch in öffentlichen Netzen einsetzen lassen, da sich die prinzipiellen Überwachungsprobleme stark ähneln.

1.3 Übersicht über die Arbeit

Nachdem in den Themenbereich der Rechnernetze und deren Management bereits eingeführt wurde, faßt Kapitel 2 die Grundlagen der Netzüberwachung zusammen. Darin eingeschlossen ist eine allgemeine Einführung in das Netzmanagement. Es wird auf das Messen in Kommunikationsnetzen eingegangen, und es werden die dabei anzutreffenden Meßprinzipien und Meßsysteme vorgestellt. Schließlich wird auf die verschiedenen Konzepte für die Überwachung innerhalb des Netzmanagements eingegangen. Abschließend werden bestehende Ansätze für Überwachungssysteme kurz vorgestellt.

In Kapitel 3 werden sowohl die Ziele einer Netzüberwachung beleuchtet, als auch die Anforderungen, die an ein Überwachungssystem gestellt werden, betrachtet. Es wird ein neues Konzept beschrieben, welches erlaubt, verschiedene Überwachungsvorgänge mit unterschiedlichen Meßsystemen in einem Überwachungssystem zu integrieren. Dieses Konzept legt besonderen Wert auf die Realisierbarkeit und eine hohe Leistungsfähigkeit im realen Einsatz.

Kapitel 4 befaßt sich mit der Umsetzung des in Kapitel 3 vorgestellten Konzeptes in ein Überwachungssystem. Einen Schwerpunkt bildet dabei die Systemarchitektur, die speziell auf die besonderen Anforderungen an Überwachungssysteme zugeschnitten ist. Abschließend wird auf den realisierten Prototyp des Systems eingegangen.

Trotz der Implementierung des Systems können fundierte Aussagen über das Zeitverhalten bei einem Einsatz in größeren Netzen nur sehr unzureichend gemacht werden, da realistische Lastsituationen nur mit sehr großem Aufwand erzeugt werden können. Es wurde deshalb eine simulative Untersuchung durchgeführt. Die Vorgehensweise und die Ergebnisse sind in Kapitel 5 beschrieben.

Die Arbeit schließt in Kapitel 6 mit einer zusammenfassenden Darstellung und einer Bewertung des Konzeptes sowie einem Ausblick.

Kapitel 2

Grundlagen der Netzüberwachung

2.1 Einführung in das Netzmanagement

Die Definition des Begriffs Netzmanagement gestaltet sich schwierig, wenn man den gesamten Komplex erfassen will. In [32] wird es definiert als die Gesamtheit der Vorkehrungen und Aktivitäten zur Sicherstellung des effektiven und effizienten Einsatzes eines Kommunikationssystems. Dies ist ein vielschichtiger Prozeß, der mit der Planung, der Installation und dem Betrieb eines Netzes einhergeht. Analog zur offenen Rechnerkommunikation, die eine Standardisierung der Kommunikationsdienste und Protokolle voraussetzt, erfordert ein integriertes Management heterogener Netze ebenfalls allgemein anerkannte Normen. Die Standardisierung einer Netzmanagementarchitektur leistet einen wesentlichen Beitrag zur Strukturierung des Problemkreises. Sie muß neben den Kommunikationsaspekten auch die sehr wichtigen Organisations-, Informations- und Funktionsaspekte abdecken und entsprechende Modelle bereitstellen [33]. Es wird unterschieden zwischen den Managementaspekten, die einzelne Kommunikationsschichten betreffen (*Layer Management*) oder Gegenstand der Kommunikationsprotokolle sind (*Layer Operation*), und dem *Systems Management*, welches die Systeme im Netz als Ganzes betrachtet und Gegenstand dieser Einführung ist.

Die Aufgabengebiete werden in fünf Funktionsbereiche strukturiert, welche als *Systems Management Functional Areas* (SMFA) genormt sind [49]. Sie lauten:

- Configuration Management (Konfigurationsmanagement),
- Fault Management (Fehlermanagement),
- Performance Management (Leistungsmanagement),
- Security Management (Sicherheitsmanagement),
- Accounting Management (Abrechnungsmanagement).

Die Grundstruktur des Systems Managements basiert auf einem Managementsystem (*Manager*) als einer zentralen Instanz und Managementagenten, welche in den im Netz befindlichen Systemen lokalisiert sind (*Agents*). Das Managementsystem kann wiederum eine verteilte Applikation sein. Die Agents sind für die systemabhängige Umsetzung der Operationen des Managementsystems verantwortlich. Manager und Agent kommunizierten miteinander über den Dienst der *Systems Management Application Entity* (SMAE). Das Managementsystem stellt dem menschlichen Netzadministrator eine Managementkonsole zur Verfügung, über die idealerweise sämtliche Managementaktivitäten rechnerunterstützt abgewickelt werden können. Die managementrelevanten Aspekte des Netzes und dessen Komponenten werden als sogenannte *Managed Objects* modelliert, auf die alle Managementoperationen zielen. Alle Managed Objects zusammen bilden eine konzeptionelle, über das Netz verteilte Datenbank, die *Management Information Base* (MIB). Managed Objects sind selbst aktiv und können mit Hilfe von Notifikationen das Managementsystem spontan über besondere Ereignisse informieren. Die Funktionalität der Managed Objects wird durch die Agents erbracht, die jeweils einen Teil der MIB realisieren. Bild 2.1 zeigt die abstrakte Sicht auf das Netzmanagement und deren Abbildung auf reale Systeme und Managementapplikationen.

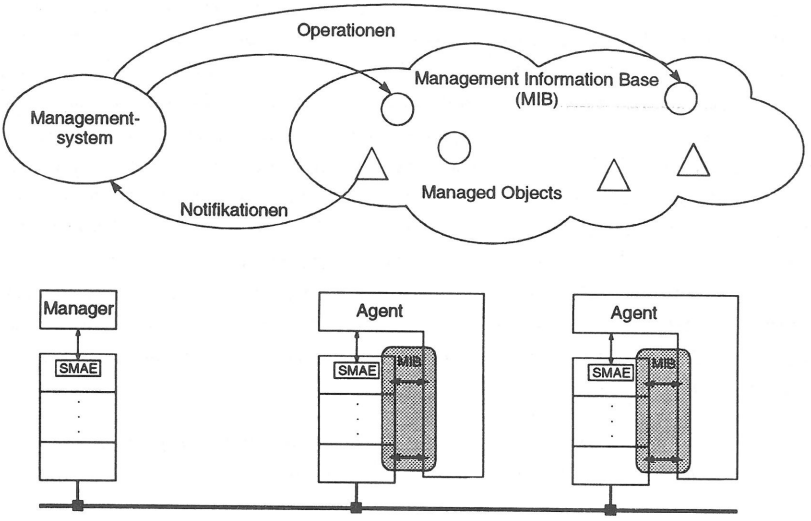


Bild 2.1: Basisarchitektur des Netzmanagements

2.1.1 Die OSI-Netzmanagementarchitektur

In diesem Kapitel werden die wichtigsten Gesichtspunkte der OSI-Netzmanagementarchitektur kurz zusammengefaßt. Eine ausführlichere Einführung findet sich in [32] und [126]. Die Be-

schreibung des OSI Systems Managements erfolgt durch vier verschiedene Modelle. Das *Organisationsmodell* befaßt sich mit den Rollen der verteilten Instanzen des Netzmanagements [62]. Neben der Definition von Manager und Agent wird ein Domänenkonzept festgelegt, welches über die Spezifikation verschiedener Zuständigkeitsbereiche erlaubt, daß mehrere Managementsysteme miteinander kooperieren.

Die Modellierung der Managementinformationen erfolgt objektorientiert und wird durch das *Informationsmodell* beschrieben [71]. Die objektorientierten Paradigmen Datenkapselung, Klassenkonzept, Vererbung, Methoden bzw. Nachrichten, sowie Polymorphie [9], [96] sind im Informationsmodell alle enthalten. Von außen sichtbar sind bei einem Managed Object folgende Elemente:

- Attribute (Daten eines Objektes),
- Meldungen (Notifikationen, die dieses Objekt als Nachricht aussenden kann) und
- Aktionen (Operationen, die ein Objekt auf Anforderung ausführt).

Die interne Darstellung der Attribute, die Mechanismen, die das Aussenden der Meldungen kontrollieren, und die Realisierung der Aktionen sind durch das Managed Object gekapselt. Die Klasse, zu der ein Objekt gehört, legt dessen Eigenschaften und die Schnittstelle zum Managementsystem fest. Der Standard *Guidelines for the Definition of Managed Objects* (GDMO) [72] beschreibt eine Notation und Schablonen für die Definition der Objektklassen und ihrer Elemente. Zusammengehörige Attribute, Meldungen und Aktionen werden zu Gruppen (*packages*) zusammengefaßt und sind als Einheit in einer oder mehreren Objektklassen enthalten. Die Bedeutung der Attribute und Aktionen sowie die Umstände, unter denen eine Meldung ausgesendet wird, sind in Form einer textuellen Verhaltensbeschreibung des Elementes (*behaviour*) definiert.

Man unterscheidet zwei Arten von Objektklassen. *Resource Managed Objects* verkörpern real existierende Kommunikationsressourcen, wie Protokollinstanzen oder Anwendungsprozesse. Ganze Systeme werden als Resource Managed Objects der Klasse *System* modelliert. Mit Hilfe von *Support Managed Objects* werden Managementfunktionen modelliert, die den Manager unterstützen, jedoch keinen direkten Bezug zu einer Kommunikationsressource besitzen und deshalb nicht über Aktionen eines Resource Managed Objects modelliert werden können. Beispiele hierfür sind in Kapitel 2.5 zu finden.

Objektklassen und Objektelemente müssen weltweit eindeutig benannt werden, damit in voneinander unabhängigen Implementierungen dieselbe Definition zugrundegelegt wird. Sie werden mit Hilfe des abstrakten Datentyps des *Object Identifiers*, der in ASN.1 (*Abstract Syntax Notation One)* [76] definiert ist und als eine Folge von natürlichen Zahlen dargestellt werden kann, benannt. Die Zahlenfolge beschreibt den Weg in einer hierarchischen Baumstruktur, dem *Registration Tree*. Dieser Baum enthält alle international eindeutigen Object Identifier und

wird von Normungsgremien hierarchisch verwaltet. Es existieren jedoch auch Teilbäume, die einzelnen Herstellern zugeordnet oder als Experimentalbereich deklariert sind.

Die Strukturierung der MIB erfolgt über Enthaltenseins-Beziehungen zwischen Managed Objects. Dabei muß eine solche Beziehung nicht notwendigerweise bedeuten, daß eine Ressource physikalisch eine andere enthält, wie dies z.B. bei einem Rechnersystem und einer Protokollinstanz der Fall ist. Letztendlich stehen alle Objekte in einer hierarchischen Beziehung zu jeweils genau einem übergeordneten Objekt und bilden so einen Baum von Objekten, den *Containment Tree*. Die erste Ebene unterhalb der Wurzel wird in diesem Baum von Objekten der Klasse System eingenommen. Die Enthaltenseins-Beziehung wird für die Benennung der Objektinstanzen herangezogen. Ein Objekt wird durch den *Relative Distinguished Name* (RDN) relativ zu seinem übergeordneten Objekt eindeutig bezeichnet. Der komplette, eindeutige Objektname (*Distinguished Name*) ergibt sich aus der Verkettung der RDNs entlang des Weges im Containment Tree von der Wurzel bis zum betreffenden Objekt. Der RDN wird bei der Objekterzeugung lokal vergeben.

Ein wichtiger Aspekt des OSI-Informationsmodells ist der dynamische Charakter der Managed Objects und damit des Containment Trees. So führt z.B. das Einschalten eines Rechnersystems zu der Entstehung eines neuen Systemobjektes in der MIB mit allen darin enthaltenen Objekten. Der Abbruch einer Transportverbindung führt entsprechend zur Zerstörung des Verbindungsobjektes. Die meisten Resource Objects werden von dem zugehörigen Agent erzeugt bzw. gelöscht, und ihre Existenz ist mit der des Systems gekoppelt. Es ist jedoch denkbar, daß ein Managementsystem mit der Löschung eines Verbindungsobjektes die zugehörige Kommunikationsverbindung abbricht. Das Managementsystem kann die Managementaktivitäten innerhalb eines Systems sehr flexibel steuern, indem es entsprechende Support Managed Objects erzeugt bzw. löscht. Gleichfalls kann der Agent über diese Objekte seine eigenen Funktionen dem Manager präsentieren.

Aus dem Vererbungsprinzip bei der Definition der Objektklassen ergibt sich eine dritte Hierarchie. Alle Klassen von Objekten sind direkt oder indirekt von der Oberklasse *Top* abgeleitet, von der sie grundlegende Eigenschaften erben. Die Vererbung erfolgt strikt, was bedeutet, daß eine Klasse die Elemente aller direkten und indirekten Oberklassen enthalten und unterstützen muß und keines ausblenden darf. Mehrfache Vererbung, bei der eine Klasse von verschiedenen Klassen abgeleitet wird, ist möglich. Die Ableitungshierarchie läßt sich in einem Vererbungsbaum (*Inheritance Tree*) darstellen. Ableitung bedeutet die Spezialisierung oder Verfeinerung einer allgemeineren Klasse und ist bei Resource Managed Objects nur sehr begrenzt anwendbar. Bei Support Managed Objects hingegen ist die Vererbung ein sehr leistungsfähiges Mittel, Funktionen zu strukturieren und gemäß der Methodik der objektorientierten Analyse auf verschiedenen Abstraktionsebenen zu beschreiben.

Die Interaktionen zwischen Managementsystem und Managed Objects erfolgen über Nachrichten, die sich auf das Lesen und Modifizieren von Attributen, das Anstoßen von Aktionen und das Versenden von Meldungen an das Managementsystem beschränken. Das *Kommunikationsmodell* für das Systems Management beschreibt die Dienste und Protokolle, die für die Übertragung dieser Nachrichten benötigt werden. Die Kommunikation basiert auf einem vollständigen OSI-Protokollstack mit sieben Schichten. Die Struktur der Verarbeitungsinstanz ist in Bild 2.2 dargestellt.

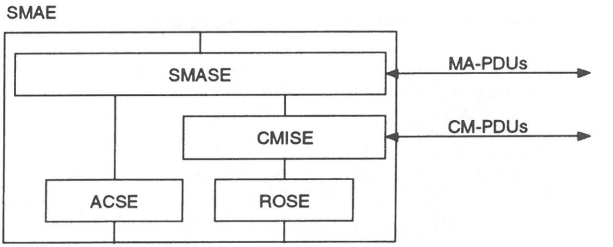


Bild 2.2: Struktur der Anwendungsinstanz für das Systems Management

Den Anwendungsprozessen steht der Dienst des *Systems Management Application Service Element* (SMASE) zur Verfügung. Dieser Dienst ist verbindungsorientiert und baut für die Verbindungssteuerung auf dem allgemeinen *Association Control Service Element* (ACSE) [51] auf. Die Zuordnung eines Managed Objects zu der Adresse des zugehörigen Agent-Prozesses ist Sache des Managementsystems, kann jedoch mit Hilfe des *Directory Service* [77] aus dem Systemobjekt ermittelt werden. Das Dienstelement CMISE (*Common Management Information Service Element*) [60] stellt für jeden Nachrichtentyp der Managed Objects ein Dienstprimitiv zur Verfügung und kommuniziert mit Partnerinstanzen über das *Common Management Information Protocol* (CMIP) [61] durch den Austausch von *Common Management*-PDUs (CM-PDUs). Die *Management Application*-PDUs (MA-PDUs) der SMASE werden 1:1 auf CM-PDUs abgebildet. Die Dienstprimitive M-GET und M-SET erlauben das Lesen und Setzen von Attributwerten und M-ACTION das Anstoßen von Operationen. Eine Ereignismeldung wird über das Dienstprimitiv M-EVENT-REPORT übermittelt. M-CANCEL-GET erlaubt den Abbruch einer laufenden Leseoperation. Darüber hinaus existieren zwei Dienstprimitive zum Erzeugen und Löschen von Managed Objects, M-CREATE und M-DELETE. Der Dienst des CMISE ist ein objektunabhängiger, allgemeiner Managementdienst, mit dem Methoden eines entfernten Objektes aufgerufen werden, was der Ausführung von entfernten Operationen entspricht und deshalb auf dem allgemeinen Dienstelement ROSE (*Remote Operations Service Element*) [57] aufbaut.

Die Dienstprimitive M-GET, M-SET, M-ACTION und M-DELETE können über die Parameter scope und filter mehrere Managed Objects gleichzeitig ansprechen [138]. Bild 2.3 zeigt das

Prinzip anhand eines Ausschnittes aus einem Containment Tree [25]. Der Scope-Parameter bezieht sich auf ein Basisobjekt und beschreibt den Bereich, aus dem die Objekte auszuwählen sind. Mögliche Bereiche sind das Basisobjekt selbst, alle Objekte, die sich im Containment Tree eine bestimmte Anzahl von Ebenen unterhalb befinden oder der ganze Unterbaum ab dem Basisobjekt. Über den Filter-Parameter können Bedingungen für Attribute spezifiziert werden, die ein Objekt erfüllen muß, um ausgewählt zu werden.

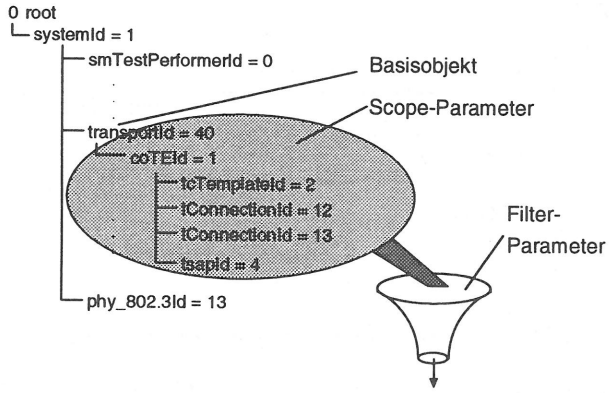


Bild 2.3: Parameter zur Objektauswahl bei CMIS-Dienstprimitiven

Das *Funktionsmodell* beschreibt, wie die Managed Objects verwendet werden sollen, um bestimmte Managementaufgaben zu erfüllen. In den entsprechenden ISO-Standards [63] werden Basisfunktionen, sogenannte *Systems Management Functions* (SMFs), genormt, die in verschiedenen Aufgabenbereichen eingesetzt werden können. Ausgehend von einem Modell, welches die Konzepte und Mechanismen der Funktion beschreibt, werden die benötigten Managed Objects aufgeführt und ihre Verwendung festgelegt. In Kapitel 2.5 sind einige für die Netzüberwachung wesentliche SMFs näher erläutert.

2.1.2 Netzmanagementstandards in lokalen Netzen

Neben der Standardisierung des OSI-Netzmanagements hat vor allem im Bereich der lokalen Netze der Managementansatz der Internet Engineering Task Force (IETF) eine breite Akzeptanz gefunden und verdrängt immer mehr firmenspezifische Managementsysteme und Protokolle [84], [85]. Dieses Managementkonzept beruht auf dem *Simple Network Management Protocol* (SNMP) [12] und wird deshalb als SNMP-Management bezeichnet. SNMP ist Teil der Protokollfamilie TCP/IP (*Transmission Control Protocol/Internet Protocol*) [19], welche die Grundlage des weltweiten Internets bildet. Das Organisationsmodell wurde weitgehend vom OSI-Management übernommen, das Informations- und Kommunikationsmodell jedoch

stark vereinfacht, um möglichst rasch und einfach implementierbar zu sein. Eine vertiefende Einführung findet sich in [33], [81], [117],[120].

Im SNMP-Management werden Informationen in Objekten abgelegt, deren Datentypen sich auf unstrukturierte Typen wie Integer-Zahlen und Strings beschränken [83]. Diese SNMP-Objekte sind mit einfachen Attributen des OSI-Managements vergleichbar. Die Klasse eines SNMP-Objektes legt den Datentyp, die Semantik und die erlaubten Zugriffsoperationen fest. Alle Objektklassen sind in öffentlich zugänglichen Dokumenten, den *Requests for Comments* (RFCs), beschrieben und in dem internationalen Registration Tree registriert. Eine SNMP-MIB besteht aus einer Menge von Objekten, deren Klassen eine gemeinsame Wurzel in diesem Registration Tree besitzen und immer als ganzes von einem SNMP-Agent unterstützt werden. Ergänzend zu der MIB-II [103], der Standard-MIB, die jeder SNMP-Agent unterstützen muß, sind verschiedene MIBs als Erweiterungen für unterschiedliche System- und Netztypen definiert worden.

Die einzige Variabilität in der Struktur einer SNMP-MIB besteht in der Möglichkeit, optionale Objekte nicht zu unterstützen oder von einer Objektklasse mehrere Instanzen anzubieten, die eine Art Liste bilden. Der Name eines Objektes ergibt sich direkt aus dessen Klassennamen und einem nachgestellten Bezeichner (Suffix), durch den sich mehrere Instanzen einer Objektklasse unterscheiden. Für einfache Objekte ist dieser Suffix immer „0". Eine Tabelle wird über mehrere zusammengehörige Klassen, die den Tabellenspalten entsprechen, realisiert. Eine Zeile ist dadurch gekennzeichnet, daß die Objekte denselben Suffix-Wert besitzen. Leider ist das SNMP-Protokoll nicht in der Lage, eine Tabelle oder Teile einer Tabelle als Gesamtheit anzusprechen. SNMP-Objekte sind immer fest einem System zugeordnet, welches einen SNMP-Agent besitzt, und werden über dessen Netzadresse angesprochen.

Das SNMP-Protokoll erlaubt das Lesen und Setzen von SNMP-Objekten und, über sogenannte *Traps*, einem SNMP-Agent einfache Meldungen unbestätigt an einen Manager zu übermitteln. Es basiert auf dem verbindungslosen, ungesicherten *User Datagram Protocol* (UDP) [110], welches sich auch auf einfachen Systemen leicht implementieren läßt. Dies bedingt jedoch, daß sich das Ergebnis einer Managementoperation nicht ändern darf, falls diese Operation, aufgrund des Verlustes einer Antwort, mehrfach ausgeführt wird. Jede SNMP-PDU enthält als Sicherheitsmechanismus einen *community string*, eine Art Paßwort, mit dem Zugriffsrechte auf verschiedene Bereiche einer MIB verknüpft werden können. Mit Hilfe des GET-NEXT-Requests ist es möglich, Objekte, deren Instanzname nicht bekannt ist, zu lesen oder nacheinander auf ganze Objektgruppen zuzugreifen. Dies ist die einzige Möglichkeit, Tabellen auszulesen. Bild 2.4 zeigt die Raum-Zeit-Diagramme der prinzipiellen Kommunikationszenarien bei SNMP.

Trotz der Tatsache, daß eine SNMP-Unterstützung in fast allen neuen Netzkomponenten enthalten ist, wird es überwiegend nur zur Abfrage von Informationen benützt [130]. Dies liegt

vor allem an den ungenügenden Sicherheitsmechanismen, die unbefugte und ungewollte Veränderungen an der Konfiguration eines Systems verhindern sollen. Hier schafft die Nachfolgeversion SNMPv2 [13]-[16] durch Authentisierungsfunktionen und Verschlüsselungskonzepte Abhilfe. Andere Erweiterungen erleichtern den Transfer von großen Tabellen und ermöglichen die Manager-Manager-Kommunikation. Sie erhöhen jedoch auch die Komplexität und damit den Speicher- und Rechenbedarf der Implementierungen, so daß viel von der Einfachheit dieses Ansatzes verlorengeht.

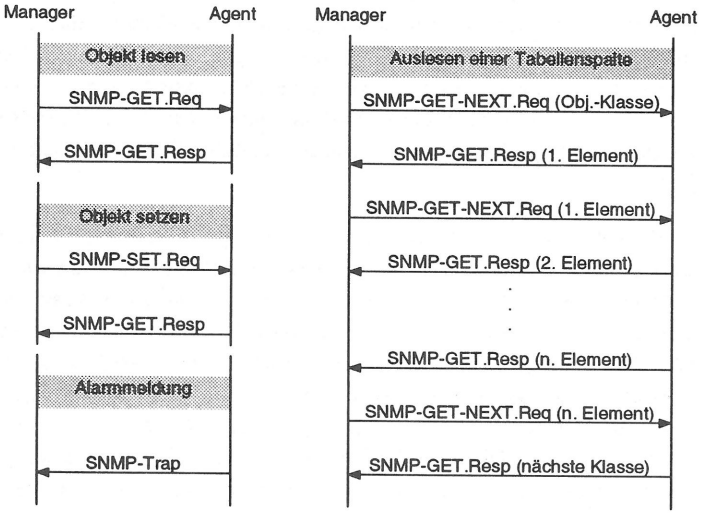


Bild 2.4: Raum-Zeit-Diagramme des Simple Network Management Protokolls

2.1.3 Netzmanagement in öffentlichen Netzen

In den öffentlichen Kommunikationsnetzen verlagert sich der Schwerpunkt von der reinen Leitungs- bzw. Paketvermittlung immer mehr zu dem Anbieten von komplexen, softwarebasierten Diensten, bei dem der eigentliche Datentransfer in den Hintergrund rückt. Diese dienstorientierten Netze werden auch als Intelligente Netze (*Intelligent Networks*, IN) bezeichnet. Mit der immer größer werdenden Anzahl von intelligenten Netzkomponenten[1], die miteinander kommunizieren, erfährt das Management dieser Systeme und der Kommunika-

[1] Der Begriff *Intelligenz* soll hier in einer erweiterten Bedeutung, wie sie im englischsprachigen Raum gebräuchlich ist, verstanden werden und mit Information und Informationsverarbeitung gleichgesetzt sein. Es ist im Englischen kein Widerspruch, algorithmisch ablaufende Programme komplexerer Natur, die ein flexibles Verhalten zeigen, als „intelligent" zu bezeichnen.

tionsinfrastruktur eine essentielle Bedeutung. Die bisherigen Signalisiernetze waren so ausgelegt, daß sie sich selbst im Rahmen des Schichtenmanagements verwalteten [75]. Durch die Heterogenität der intelligenten Komponenten ist eine standardisierte offene Kommunikationsarchitektur [74] und ein standardisiertes Netzmanagement erforderlich [73]. *Telecommunications Management Network* (TMN) bezeichnet sowohl ein spezielles Netz, welches dem Nutzdatennetz und dem Signalisiernetz speziell zum Zwecke des Managements überlagert ist, als auch eine Managementarchitektur für Telekommunikationsnetze. Bild 2.5 stellt die Funktionsblöcke und Referenzpunkte des TMN mit ihren Abkürzungen dar.

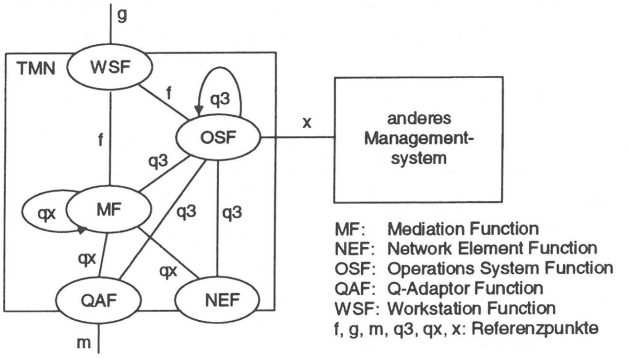


Bild 2.5: TNM-Referenzarchitektur

Die NEF verkörpert die Managementaspekte der zu verwaltenden Elemente des Kommunikationsnetzes (*Network Elements*) und entspricht den Agents des OSI Systems Management. Netzelemente mit nicht TMN-konformen Schnittstellen können über die QAF in das TMN (q-Schnittstelle) integriert werden. Die eigentlichen Managementoperationen führt die OSF aus, die damit einem Manager entspricht. Statt einer zentralen Managerinstanz kooperieren mehrere OSFs miteinander. Die Netzmanagementkonsole, also der Zugang zum TMN, wird über die WSF realisiert. Die OSF kommuniziert mit den NEFs über die q3-Schnittstelle, die CMIS/CMIP entspricht. In heterogenen Umgebungen wird die MF als Mittler zwischen der q3-Schnittstelle und anderen TMN-Schnittstellen (qx) eingesetzt. Vergleichbar mit dem Management heterogener lokaler Netze [114], kann die MF als sogenannter Proxy-Agent fungieren, indem sie die Informationsmodelle ineinander überführt [10], oder eine gewisse Managementfunktionalität besitzen (Element Manager) und damit die OSF entlasten.

2.2 Einführung in die Netzüberwachung

Von Rechnernetzen wird eine außerordentlich hohe Verfügbarkeit und Zuverlässigkeit gefordert. Dies kann nur durch laufende Beobachtung während des Betriebs erreicht werden. Wich-

tig ist die frühzeitige Erkennung von Überlastfällen und Fehlersituationen, um Ausfallzeiten kurz zu halten. Die Netzüberwachung ist Teil des Netzmanagements und betrifft alle Funktionsbereiche. Jeder Funktionsbereich stellt jedoch eigene Anforderungen. So ist die Erfassung des aktuellen Netzzustandes, wie er auf Managementkonsolen üblicherweise angezeigt wird, dem Configuration Management zuzurechnen, während das Erkennen von Fehlfunktionen und deren Lokalisation im Fault Management benötigt wird. Für das Performance Management gilt es, statistische Verkehrsdaten zu sammeln und für spätere Langzeit- und Trendanalysen zu speichern. Alles in allem treffen auf die Netzüberwachung eine Menge von Anforderungen, die schwer vorhergesehen werden können. Ein Überwachungskonzept muß deshalb so flexibel sein, daß es an verschiedene Managementstrategien angepaßt werden kann. Seine prinzipielle Struktur zeigt Bild 2.6.

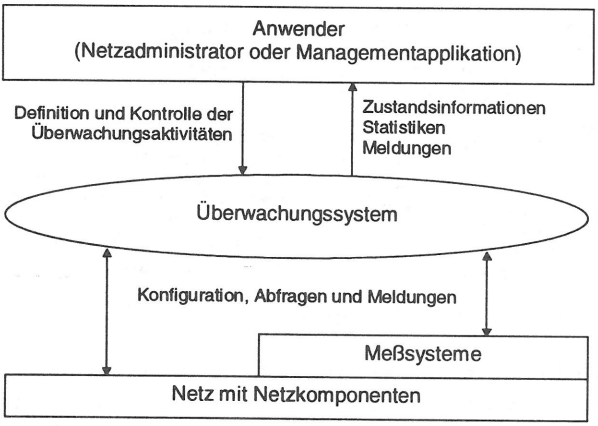


Bild 2.6: Prinzipielle Struktur der Netzüberwachung

Die Basis der Netzüberwachung bilden Daten über den Zustand und die Vorgänge im Netz. Teile dieser Daten werden von den Netzkomponenten selbst erfaßt und können über eine Netzmanagementschnittstelle abgefragt werden. Ergänzend dazu werden stationäre Meßsysteme eingesetzt, die speziell passive Komponenten und das Netz selbst überwachen. Netzkomponenten und Meßsysteme können als verteilte Sensoren betrachtet werden. Das Überwachungssystem stellt das Bindeglied zwischen den Sensoren und dem Anwender dar. Es ist zuständig für die Koordination der Überwachungsaktivitäten, deren Durchführung mit Hilfe der Sensoren sowie die Aufbereitung der gewonnenen Informationen. Für die Übermittlung der Sensordaten kommen drei Verfahren in Frage [20]:

- die einfache Abfrage,
- die zyklische Abfrage (Polling) und
- Ereignismeldungen bzw. Alarme.

Häufig erfordert der Einsatz der Sensoren für die Überwachung zusätzlich eine entsprechende Konfiguration. Zwar ist für die Kommunikation zwischen Sensor und Überwachungssystem die einheitliche Verwendung standardisierter Managementprotokolle wünschenswert, jedoch sind vor allem ältere Netzkomponenten und Meßsysteme nur über proprietäre Protokolle ansprechbar.

Aus der Managementstrategie eines Netzes ergeben sich Überwachungsaktivitäten, die festlegen, welche Daten erfaßt werden, wie Schwellwerte für Alarme einzustellen sind und welche Informationen wie angezeigt bzw. gespeichert werden. Diese Aktivitäten sind jedoch stark von der verwendeten Netztechnologie, dem Zugriffsprotokoll, der Netzgröße, der Topologie, sowie installationsindividuellen Parametern abhängig und müssen in der Regel von einem menschlichen Netzadministrator definiert werden. In bestimmten Bereichen können jedoch auch Netzmanagementapplikationen solche Aktivitäten definieren und auswerten. Beispiele hierfür sind ein Expertensystem für die Netzdiagnose [128] und eine Applikation zur Überprüfung der Konsistenz von Routing-Tabellen [78].

2.3 Meßtechnik in Kommunikationsnetzen

In der Kommunikationstechnik umspannen Meßsysteme einen weiten Bereich. Da eine ausführliche Behandlung aller Meßtechnologien und -prinzipien den Rahmen dieser Arbeit sprengen würde, soll hier hauptsächlich auf Meßsysteme, die in lokalen Netzen eingesetzt werden, eingegangen werden. Die nachfolgend beschriebenen Systeme und Konzepte sind jedoch, zumindest teilweise, auch in Weitverkehrsnetzen anzutreffen.

2.3.1 Klassifizierung der Meßsysteme

Meßsysteme, die in Rechnernetzen eingesetzt werden, sind vielfältig in dem, was sie messen und wie sie es messen. In Bild 2.7 sind einige Vertreter in Produktkategorien einsortiert. Für eine Klassifizierung hinsichtlich der Meßfunktionen bieten sich die Schichten des OSI-Referenzmodells an.

Meßgeräte für die *Bitübertragungsschicht* sind vor allem bei der Netzinstallation erforderlich. Sie erfassen die Übertragungseigenschaften des Mediums und sichern einen guten Kommunikationskanal. Bei Bus-Topologien spielen vor allem Time-Domain Reflektionsmeßgeräte [8] eine große Rolle, da mit ihnen schon leichte Beschädigungen an den Kabeln und schlechte Verbindungen lokalisiert werden können, die im Betrieb die Leistungsfähigkeit des Netzes

drastisch reduzieren. Eine Messung der Signalgüte während des Betriebs [104] gibt Aufschluß über die korrekte Funktion der Übertragungseinrichtungen.

Bei vielen Netzen ist auf dem Medium eine Überlagerung der Verkehrsströme unterschiedlicher Kommunikationsprotokolle anzutreffen. Ihr gegenseitiger Einfluß kann nur auf der allen Protokollfamilien gemeinsamen *Sicherungsschicht*, der Schicht 2, erfaßt werden. Einige Kenngrößen dieser Schicht sind jedoch spezifisch für ein Medienzugriffsprotokoll, wie Kollisionsraten oder Token-Umlaufzeiten. Andere Größen müssen in verschiedenen Netzen unterschiedlich interpretiert werden. So ist z.B. die momentane Datenrate in Relation zur Bitrate des Netzes zu setzen. Eine Überwachung der Sicherungsschicht bezieht sich auf den Gesamtverkehr, der über Protokollanalysatoren und Netzmonitore statistisch erfaßt wird, die in Kapitel 2.3.4 genauer behandelt werden. Bei der Überwachung ist zu beachten, daß Netze häufig zur Lasttrennung in Teilnetze aufgeteilt werden, die über Brücken und Router miteinander verbunden sind. In diesem Fall müssen die Kenngrößen in jedem Teilnetz separat erfaßt werden. Für die Erkennung von Fehlfunktionen an den Koppelelementen sind Meßgeräte mit zwei Netzanschlüssen notwendig, damit diese Systeme „in die Zange" genommen werden können [106].

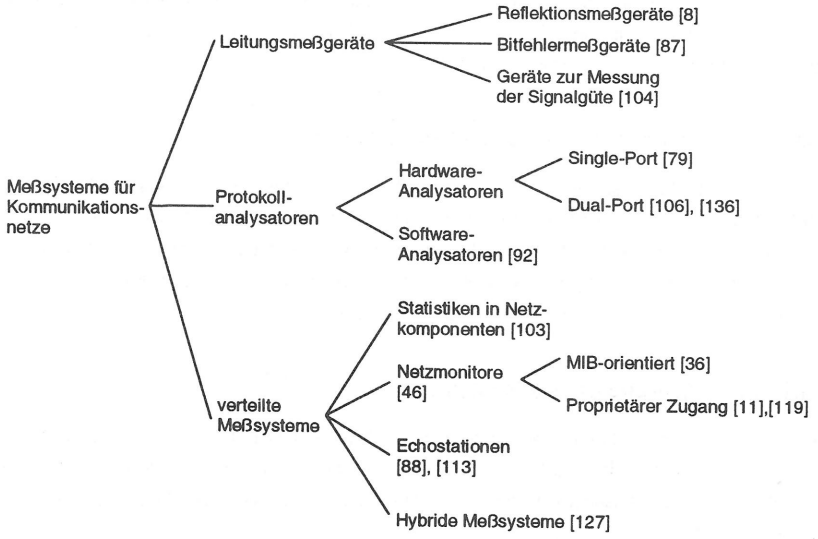


Bild 2.7: Meßsysteme für Kommunikationsnetze

Oberhalb der Schicht 2 können viele Meßfunktionen von den Netzkomponenten selbst übernommen werden, sofern sie genügend Rechenkapazität und eine Managementschnittstelle besitzen. Die Daten, die diese Komponenten einer Überwachung zur Verfügung stellen, beschränken sich jedoch auf die Komponente selbst und deren Betriebsmittel für die Kommu-

nikation. Neben Zustands- und Konfigurationsinformationen wird von den Stationen häufig der ankommende und abgehende Verkehr in Form von Paketzählern erfaßt. Hinzu kommen Zähler für fehlerhafte Pakete und Protokollfehler. Besitzt eine wichtige Netzkomponente keine Managementschnittstelle, kann diese auch indirekt durch ein Meßsystem überwacht werden, welches den Verkehr von und zu dieser Station überwacht [90].

Eine andere Klassifikation der Meßsysteme ist die Einteilung in:

- passive Systeme (z.B. Netzmonitore),
- Lastgeneratoren und
- Testsysteme.

Passive Systeme zeichnen sich dadurch aus, daß sie das Meßobjekt nur beobachten und nicht beeinflussen. Mit Hilfe von Lastgeneratoren wird künstlicher Netzverkehr erzeugt und dadurch eine bestimmte Lastsituation im Netz geschaffen, vor deren Hintergrund die Messungen stattfinden. Unter Testsystemen versteht man Meßsysteme, die im Rahmen einer Messung (Test) das Meßobjekt beeinflussen. Dazu zählen u.a. Leitungsmeßgeräte und Systeme für den Conformance-Test bzw. Performance-Test, welche die Meßobjekte in einer speziellen Umgebung untersuchen. Testsysteme für die Netzüberwachung überprüfen bestimmte Eigenschaften des Netzes im Betrieb, indem sie aktiv kommunizieren. Eine wichtige Eigenschaft ist die Erreichbarkeit der einzelnen Endsysteme auf verschiedenen Schichten. Sie wird ermittelt, indem das Testsystem einen besonderen Testrahmen aussendet, den die Zielstation unverändert an die Absenderadresse zurücksendet (Echotest). In verschiedenen Protokollspezifikationen sind solche Testrahmen enthalten [52], [111]. Für die Durchführung der Echotests existieren spezielle Echostationen [88], [113], die selbst wiederum als verläßliche Partner in einem Echotest fungieren können.

2.3.2 Meßprinzipien

Messungen in Netzen sind geprägt von der geographischen Ausdehnung des Meßobjekts. Die Netzkenngrößen sind ortsabhängig und müssen deshalb durch verteilte Sensoren erfaßt werden. Die Ortsabhängigkeit ergibt sich aus der Strukturierung des Netzes in Teilnetze und Netzsegmente, die jeweils eine eigene Charakteristik besitzen. Bei Netzen mit einer Bus-Topologie und wahlfreiem Zugriff spielen innerhalb eines Netzsegmentes auch noch Laufzeiten und Überlagerungseffekte eine Rolle. Neben der Erfassung der Netzkenngrößen ergeben sich für die Netzüberwachung folgende Meßaufgaben:

- Ermittlung des Verkehrsaufkommens zwischen den Endsystemen (Verkehrsmatrix),
- Erkennen und Lokalisieren von Informationsverlusten,
- Feststellung der Erreichbarkeit eines Endsystems (Echotest),
- Laufzeitbestimmungen,

- Verfolgung des Weges eines Paketes durch das Netz und
- Erfassung des Gesamtzustands des Netzes.

Diese Aufstellung erhebt keinen Anspruch auf Vollständigkeit. Sie führt jedoch zu den wesentlichen Prinzipien, die bei Messungen in Netzen Anwendung finden:

- Beobachtung,
- Aufzeichnung der Netzaktivitäten,
- Zweipunktmessungen,
- Round-Trip-Messungen,
- Verwendung von Testfunktionalität innerhalb der Kommunikationsprotokolle.

Die meisten Netzkenngrößen können aus der zeitlichen und inhaltlichen Beobachtung aller Pakete auf dem Übertragungsmedium abgeleitet werden. Dies ist die Domäne der Protokollanalysatoren und Netzmonitore. Mit ihnen können auch Verkehrsmatrizen aufgestellt werden. In großen Netzen mit vielen Endsystemen sind für Verkehrsmatrizen effiziente Speicherungsmethoden erforderlich, und es ergibt sich das Problem der Darstellung [105]. Problematisch ist es, gelegentliche Paketverluste zu erkennen und die Ursache dafür zu ermitteln [45]. Zwar werden diese Verluste in den höheren Protokollschichten erkannt, jedoch erst nach einiger Zeit. Deshalb ist es häufig wichtig, die Vorgeschichte eines Ereignisses in Verkehrsaufzeichnungen nachvollziehen zu können.

Bei Laufzeitmessungen zwischen zwei Punkten besteht das Problem darin, daß Zeitpunkte an räumlich entfernten Punkten zueinander in Beziehung gesetzt werden müssen. In [137] und [140] sind hierfür verschiedene Verfahren beschrieben. Für die Netzüberwachung kann jedoch häufig darauf verzichtet werden, indem die Testpakete an der Zielstation gespiegelt werden, wodurch beim Absender die doppelte Verzögerungszeit gemessen werden kann. Zur Durchführung solcher *Round-Trip*-Messungen können die für Echotests vorgesehenen Testpakete verwendet werden.

Die Überprüfung der Erreichbarkeit von Stationen in anderen Teilnetzen schließt die Koppelelemente in den Test mit ein. Notwendig ist hierfür ein durchgängiger Kommunikationskanal und die richtige Funktion der Adressierungs- und Routingmechanismen. Die Verfolgung des Weges bestimmter Pakete im Netz gestaltet sich mit verteilten Netzmonitoren sehr schwierig, da die erfaßten Daten zeitlich eng korreliert werden müssen und das Datenaufkommen hoch ist. Sinnvoller ist es, im Protokoll der Vermittlungsschicht eine entsprechende Testfunktionalität hinzuzufügen, beziehungsweise existierende Mechanismen geschickt auszunutzen. Ein schönes Beispiel ist das Programm traceroute [132], mit dem in IP-Netzen der Weg eines Paketes ermittelt werden kann. Das Verfahren basiert darauf, daß im IP, zur Verminderung des Effektes von Schleifen in Routing-Tabellen, Pakete nur eine begrenzte Anzahl von Routern überqueren dürfen. Dies wird durch den *Time-to-live*-Zähler im IP-Paketkopf gewährleistet.

Dieser Zähler wird von jedem IP-Router dekrementiert und bei einem Zählerstand von Null wird das Paket verworfen, wobei dem Absender eine entsprechende Fehlermeldung zugestellt wird. Traceroute sendet nun ein IP-Echo-Paket an eine Station mit einem Time-to-live-Zähler von 1 und erhält vom ersten Router eine Meldung, daß das Paket verworfen wurde. Daraufhin wird das Paket so lange wiederholt, wobei der Zähler immer um eins inkrementiert wird, bis das Paket normal beantwortet wird. Aus den erhaltenen Fehlermeldungen können die Router und damit der Weg des Paketes bestimmt werden.

Die Erfassung des Gesamtzustandes eines Netzes besteht aus einem Kompromiß zwischen einer möglichst aktuellen Statusinformation über alle Komponenten und der Minimierung der daraus resultierenden Netzbelastung. Ermittelt wird dieser Gesamtzustand über periodisch durchgeführte Abfragen. Dies kann zentral durch das Überwachungssystem erfolgen oder dezentral durch Meßsysteme, die als sogenannte Element Manager fungieren [94].

2.3.3 Erfassung von Fehlern auf den unteren Schichten

Statistiken über die Ursachen von Netzausfällen zeigen, daß Probleme auf den unteren Schichten im OSI-Referenzmodell, vor allem Störungen im Bereich der Kabel und Netzanschlüsse den Hauptteil ausmachen. Dabei sind die Fehler, die auftreten können, von der Topologie des Netzes und dem Medienzugriffsverfahren abhängig. Nachfolgend werden verschiedene Netztypen auf ihre spezifischen Probleme und Überwachungsmöglichkeiten hin untersucht.

2.3.3.1 Token-Ring-Netze

Netze mit Ring-Topologie besitzen ein Medienzugriffsverfahren, welches, abgesehen von getakteten Ringen, auf dem Token-Verfahren beruht [97]. Verbreitete Vertreter sind Netze nach dem Standard ISO 8802/5 (Token Ring) [55] und FDDI (Fiber Distributed Data Interface) [58]. Sie bieten eine hohe Übertragungssicherheit, da die Signale von jeder angeschlossenen Station regeneriert werden und jedes Paket von der sendenden Station auf der anderen Seite des Ringes wieder empfangen und auf Korrektheit geprüft wird. Dadurch werden Übertragungsfehler vom Sender erkannt und Paketverluste durch eine wiederholte Übertragung verhindert. Setzt man voraus, daß alle Stationen Übertragungsfehler beim Senden an das Netzmanagement melden, bedarf es bei Token-Ringsystemen keiner besonderen Überwachung der Bitübertragungsschicht. Treten jedoch Störungen auf, sind diese nur sehr schwer zu lokalisieren. Es müssen alle Stationen aus den Ring entfernt werden, das Medium überprüft und dann nacheinander alle Stationen wieder angeschlossen werden. FDDI besitzt eine Doppelring-Struktur, die es erlaubt, beim Ausfall eines Übertragungsabschnittes, durch Umkonfiguration den Betrieb aufrecht zu erhalten. Fehlfunktionen auf Medienzugangsschicht (Schicht 2a), wie Token-Verlust, werden durch schichtinterne Managementmechanismen behandelt.

2.3.3.2 Busorientierte Netze

Netze, deren Medium als Bus ausgeführt ist, basieren überwiegend auf dem Standard ISO 8802/3 [53] bzw. Ethernet [22], und verwenden das CSMA/CD-Zugriffsprotokoll (Carrier Sense Multiple Access/Collision Detect). Im Bereich der Fertigungsautomatisierung wird auch das Token-Bus-Protokoll (ISO 8802/4) eingesetzt. Bei Bus-Topologien ist der Medienzugang rein passiv ausgeführt, d.h. in den Stationen findet keine Signalregenerierung statt. Eine sendende Station hat keine Möglichkeit, den korrekten Empfang ihres Paketes zu überwachen, so daß Übertragungsfehler zu Paketverlusten führen. Diese Übertragungsfehler können ortsabhängig sein, d.h. manche, dem Sender nahe Stationen, empfangen das Paket korrekt, während andere, weiter entferntere Stationen, es verwerfen. Um Übertragungsprobleme frühzeitig erkennen und lokalisieren zu können, empfiehlt es sich, in kritischen Umgebungen, wie z.B. in Fabrikhallen, das Medium selbst während des Betriebs mit Reflektionsmessungen auf Beschädigungen hin zu überwachen [128]. Die weite Verbreitung von sternförmigen Verkabelungsstrategien lassen dieses Problem jedoch in den Hintergrund treten (siehe Kapitel 2.3.3.3).

Eine andere häufige Fehlerquelle ist die Unterbrechung der Verbindung zum Medium. Da Stationen passiv an den Bus gekoppelt sind, wird bei CSMA/CD-Netzen die Abkopplung einer Station nur dadurch erkannt, daß diese nicht mehr erreichbar ist und selbst nicht mehr kommunizieren kann.

2.3.3.3 Sternförmige Netze

Die Neuverkabelung von Gebäuden erfolgt immer häufiger sternförmig, da dies eine größtmögliche Flexibilität bezüglich der verwendeten Netztechnologie gewährleistet. Netze mit Sternstruktur besitzen eine zentrale Einheit, die als Knoten oder *Hub* bezeichnet wird. Wird ein entsprechender Hub eingesetzt, können sowohl Token-Ring-Netze als auch CSMA/CD-Netze realisiert werden. Das eigentliche Übertragungsmedium degeneriert dabei zu einem Knoten, und der verteilte Zugriff erfolgt im Hub [35], [47]. Viele Hubs sind mit einer SNMP-Managementschnittstelle ausgestattet und somit zentral überwachbar und konfigurierbar. Sie überwachen das Medium und sind in der Lage, Endsysteme, die den Gesamtbetrieb beeinträchtigen, zu erkennen und vom restlichen Netz zu isolieren.

2.3.3.4 Getaktete Netze mit Mehrfachausnutzung des Mediums

Bei Hochgeschwindigkeitsnetzen wird das Medium meistens synchron getaktet und unidirektional betrieben. Einzelne Zeitschlitze können von Stationen für die Übertragung eines Paketes belegt werden. Manche Zugriffsprotokolle sehen nun vor, daß ein Zeitschlitz vom Empfänger wieder als frei gekennzeichnet wird und ein zweites Mal verwendet werden kann. Dieses Verfahren ist zwar im Wirkbetrieb noch nicht anzutreffen, wird jedoch bei den Zugriffsprotokollen CRMA-II (Cyclic-Reservation Multiple-Access) und DQDB (Distributed Queue Dual Bus)

diskutiert [2], [56], [115]. Für die Netzüberwachung stellt sich hierbei das Problem, daß an keiner Stelle mehr alle Pakete auf dem Netz beobachtet werden können, d.h. ein Einsatz von Netzmonitoren in herkömmlicher Form nicht mehr denkbar ist. Eine umfassende Überwachung müßte die Verkehrsparameter auf jeder Verbindung zwischen zwei Stationen erfassen, um ein vollständiges Bild über die Verkehrsströme im Netz zu erhalten. Dies ist aus Aufwandsgründen jedoch nicht möglich.

2.3.4 Netzmonitore

Protokollanalysatoren und Netzmonitore beobachten die Rahmen auf einem Netz. Protokollanalysatoren sind hauptsächlich für den Servicetechniker vor Ort für die Inbetriebnahme und Fehlersuche bestimmt und deshalb portabel. Sie lassen sich direkt über Tastatur und Bildschirm bedienen. Ihre Hauptaufgabe ist die Erfassung, Aufzeichnung und Darstellung aller Rahmen auf dem Netz oder nur derjenigen, die bestimmte Filterbedingungen erfüllen. Damit bei hoher Netzlast der Datenpuffer nicht zu schnell überläuft, erfolgt die Filterung üblicherweise in Echtzeit bei der Erfassung. Die erfaßten Rahmen werden dekodiert und auf dem Bildschirm dargestellt bzw. in einer Datei gespeichert. Darüber hinaus können Protokollanalysatoren selbst eine einstellbare Netzlast erzeugen. Zwar können diese Geräte auch die Netzkenngrößen und die Verkehrscharakteristik bestimmen, der Schwerpunkt liegt jedoch bei der Analyse der Rahmeninhalte.

Netzmonitore (*LAN-Probes*), wie sie hier betrachtet werden sollen, beschränken sich auf den stationären Einsatz in einem Netzsegment und werden zentral über das Netz gesteuert [46]. Sie sind für die Überwachung eines Netzes ausgelegt und nur bedingt für die Fehlersuche geeignet. Ein geringer Hardwareaufwand hält die Herstellungskosten niedrig und erlaubt den mehrfachen Einsatz im Netz. Die Meßmöglichkeiten reichen, je nach Ausführung, von der einfachen Erfassung der Anzahl von Paketen und Bytes in einem bestimmten Intervall, über komplexere Verkehrsstatistiken, bis hin zu der Erstellung von nach Sende- und Empfangsadresse aufgeschlüsselten Verkehrsmatrizen. Leistungsfähige Geräte besitzen die Fähigkeit, gefilterte Paketströme aufzuzeichnen und sie danach an eine zentrale Station zur Dekodierung und Darstellung zu übermitteln. Sie ähneln somit einfachen Protokollanalysatoren.

Netzmonitore besitzen eine Netzanschaltung, die im sogenannten *Promiscuous Mode* arbeitet. In dieser Betriebsart werden alle Rahmen auf dem Netz empfangen und im Hauptspeicher abgelegt, wo sie ausgewertet werden. Damit verstümmelte und fehlerhafte Rahmen unter allen Lastbedingungen empfangen werden, ist die Netzanschaltung häufig mit speziellen Hardwarekomponenten ausgestattet [112]. Es können jedoch auch normale Workstations oder Personal Computer (PCs) in diesen Modus gebracht werden und mit entsprechender Software als Protokollanalysatoren oder Netzmonitore agieren [92]. In den Arbeiten [89] und [90] wurde ge-

zeigt, daß schon auf der Basis von 16-Bit Mikroprozessoren einfache Netzmonitore preisgünstig realisiert werden können, die sich für einen verteilten Einsatz im Netz eignen.

2.3.5 Kommunikationsmechanismen und Protokolle

Für die zentrale Überwachung eines Netzes ist es unabdingbar, daß die im Netz befindlichen Meßsysteme zentral gesteuert werden können, und die Meßdaten an diese zentrale Instanz gelangen. Die hierfür notwendige Kommunikation kann sowohl, wie in Bild 2.8 dargestellt, über das zu überwachende Netz abgewickelt werden (*inband*) [137] oder über einen separaten Kommunikationskanal (*outband*) [140]. Ein separates Medium hat den Vorteil, daß bei Netzausfällen immer noch ein Zugriff auf die Meßstationen möglich ist und daß eine Beeinflussung des Nutzverkehrs durch die Kommunikation mit den Meßsystemen vermieden wird. Auf der anderen Seite stehen die Kosten für die Installation eines zweiten Netzes rein für die Überwachung. Ein erfolgreicher Zugriff auf ein Meßsystem über das zu überwachende Netz bestätigt auch implizit die Erreichbarkeit des betreffenden Netzteiles für den Nutzbetrieb.

In lokalen Netzen kommen überwiegend Inband-Lösungen zum Einsatz, da totale Netzausfälle gegenüber Teilausfällen und Leistungseinbrüchen relativ selten auftreten und den erheblichen Aufwand der Installation eines separaten Kommunikationsmediums nicht rechtfertigen. Darüber hinaus sind bei Totalausfällen sowieso Reparaturmaßnahmen vor Ort notwendig, so daß eine Ferndiagnose weniger wichtig ist. Manche Meßsysteme besitzen eine serielle Schnittstelle, die es ermöglicht, bei Bedarf über eine Telefonwählverbindung und Modems mit dem Überwachungssystem zu kommunizieren. Erfolgt die Inband-Kommunikation bereits über IP-Protokolle, empfiehlt es sich für die Kommunikation über eine serielle Verbindung, das Schicht-2-Protokoll SLIP (Serial Line Interface Protocol) [116] zu verwenden, da die darüberliegenden Schichten beibehalten werden können.

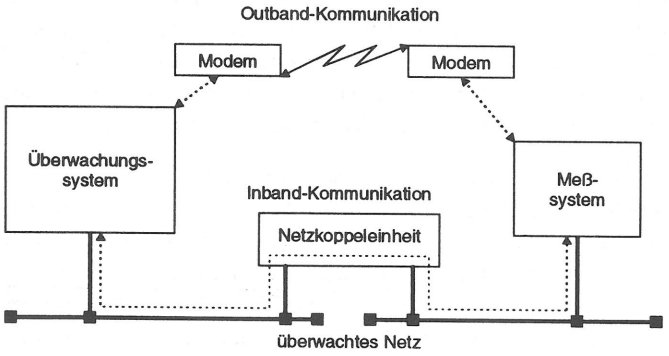


Bild 2.8: Kommunikationswege zwischen Überwachungs- und Meßsystem

Die Kommunikationsbeziehungen zwischen den verschiedenen Meßsystemen und dem Überwachungssystem können in vier Kategorien eingeteilt werden:

- Das Meßsystem sendet periodisch alle erfaßten Netzzustandsdaten an eine bestimmte Adresse. Wird eine Multicast-Adresse gewählt, können auch mehrere Überwachungssysteme diese Daten empfangen. Ein System dieser Art ist z.B. die Ethernet-Box im Pulsar-Modus der Firma RzK D. Köpke [88].
- Der Zugang erfolgt über eine *Remote-Login*-Prozedur, die es erlaubt von einer entfernten Stelle aus, die Meßkomponente direkt zu bedienen [11].
- Das Meßsystem ist über ein Netzmanagementprotokoll konfigurierbar, und die Meßergebnisse werden abgefragt bzw. als Ereignismeldungen verschickt [36].
- Die Meßstation führt Messungen aus, die über ein spezielles Meßprotokoll abgewickelt werden. Die Meßparameter werden im Meßauftrag übermittelt, und am Ende der Messung werden die Ergebnisse an die zentrale Station übertragen. [127]

Eine wichtige Rolle spielen die Kommunikationsprotokolle zwischen Meßstation und dem Überwachungssystem. Sie müssen einfach gestaltet sein, damit sie in den Meßstationen Platz finden. Sollen Meßsysteme auch über Teilnetzgrenzen hinweg gesteuert werden können, müssen sie eine Vermittlungsschicht enthalten, um Router überwinden zu können. Darüber hinaus müssen sie auch auf der zentralen Überwachungsstation verfügbar sein. Viele Meßsysteme verwenden proprietäre Protokolle. Ein Beispiel ist das Low Level Transport Protocol (LLTP), ein spezielles Meßprotokoll, welches von einem verteilten Meßsystem mit sehr unterschiedlichen Meßkomponenten verwendet wird [128]. Es ist speziell für die gesicherte Übertragung von Meßaufträgen und deren Ergebnissen ausgelegt. Häufig basiert die Kommunikation auch auf UDP, da dieses Protokoll im Rahmen der TCP/IP-Protokollfamilie auf jeder Workstation verfügbar ist. Neuere Netzmonitore unterstützen SNMP und die RMON-MIB (Remote Monitoring MIB, siehe Kapitel 2.4.1) und besitzen damit einen standardisierten Netzmanagementzugang.

Die Kommunikation des Überwachungssystems mit den Netzkomponenten und den Meßsystemen führt zu zusätzlichem Netzverkehr. Bei entsprechenden Abfrageintervallen kann dies einen erheblichen Anteil an der Gesamtkommunikation ausmachen. Ein Engpaß entsteht dabei häufig in dem Netzzugang des Überwachungssystems. Besonders kritisch sind Broadcast-Pakete in großen Teilnetzen, da sie grundsätzlich von Bridges weitergereicht werden und in allen Stationen einen erhöhten Bearbeitungsaufwand verursachen. Kommuniziert eine Überwachungsstation zyklisch mit sehr vielen Stationen im selben Teilnetz, kann dies dazu führen, daß der lokale Cache für die Schicht-2-Adressen nicht ausreicht und für jeden Informationsaustausch diese Adresse über eine Broadcast-Meldung des Address Resolution Protocols (ARP) erfragt werden muß [109].

2.4 Modellierung der von Meßsystemen erfaßten Netzdaten

Für eine erfolgreiche Kooperation zwischen dem Überwachungssystem und den Meßsystemen ist es notwendig, daß beide Seiten dasselbe Datenmodell verwenden, welches die Struktur der Netzdaten und ihre Adressierung festlegt. Das große Spektrum der Meßsysteme erfordert von dem Datenmodell eine hohe Flexibilität. Damit neue, andersartige Meßkomponenten in das Modell integriert werden können, muß es auch erweiterbar sein. Diesen Ansprüchen wird das Modell der Management Information Base gerecht. Zwar unterscheiden sich die MIBs zwischen dem OSI- und dem SNMP-Netzmanagement, beiden gemeinsam ist jedoch, daß sie systemorientiert gestaltet sind und sich sehr gut dafür eignen, Stationen im Netz mit ihren Bestandteilen hierarchisch strukturiert zu modellieren. Aus der Sicht der MIB wird das Netz als Konglomerat von Netzkomponenten betrachtet. Aus der Sicht der Netzüberwachung besitzt das Netz selbst jedoch Eigenschaften, wie die physikalische Integrität oder Verkehrsparameter, kann jedoch im Sinne der MIB nicht als System betrachtet werden, da dem Netz kein Managementagent zugeordnet werden kann. Deshalb müssen, soll die Managementarchitektur beibehalten werden, die Informationen über das Netz, wie in Bild 2.9 dargestellt, innerhalb der Meßkomponenten modelliert und abgelegt werden.

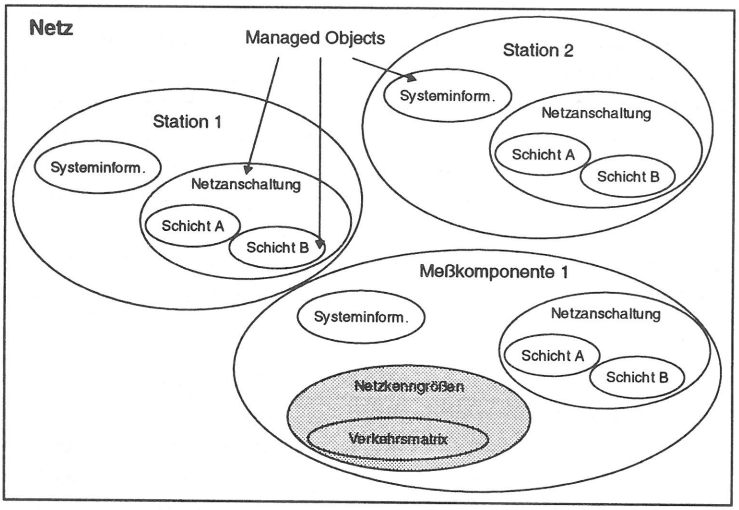


Bild 2.9: Modellierung der Informationen über das Netz innerhalb der MIB

Wenn schon die gemessenen Daten Teil der MIB der Meßkomponente sind, ist es auch sinnvoll, die Meßaktivitäten über das Beschreiben von MIB-Attributen oder das Erzeugen von neuen Objekten zu konfigurieren. Dadurch können die für das Netzmanagement bereits implementierten, standardisierten Kommunikationsprotokolle verwendet werden. Da in der Regel

nicht alle Aspekte der Meßkomponenten in den standardisierten Objektkatalogen berücksichtigt sind, werden diese Lücken über spezielle, herstellerspezifische Erweiterungen der MIB ausgeglichen. Das Überwachungssystem muß in der Lage sein, diese MIB-Erweiterungen in das eigene Objektmodell einfließen zu lassen und sie für die Anwendungen sinnvoll benutzbar zu machen.

Obwohl die Management Information Base des OSI Systems Management durch ihre flexible, objektorientierte Struktur klare Vorteile hat, ist im Bereich der lokalen Netze in den Komponenten fast ausschließlich das SNMP-Management aufgrund seiner Einfachheit implementiert. SNMP wurde schwerpunktmäßig für die Netzüberwachung entwickelt. Die MIB-II (vgl. Kapitel 2.1.2) enthält Konfigurations- und statistische Verkehrsinformationen über die Netzkomponente selbst. Sie ist auf die TCP/IP Protokollarchitektur zugeschnitten und beinhaltet unter anderem den Typ des Systems (Workstation, Router, ...), den Hersteller, die Anzahl der Netzanschaltungen sowie, für die Protokollschichten oberhalb der Vermittlungsschicht (IP), verschiedene Konfigurations- und Statistikvariablen. Für jede Netzanschaltung (Interface) sind die Schicht-2-Adresse und Paketstatistiken enthalten. Für Netzmonitore, die netzweite Informationen erfassen, wurde die Remote Monitoring Management Information Base (RMON-MIB) [135] entwickelt. In den folgenden Kapiteln werden diese MIB sowie die ihr zugrunde liegenden Mechanismen und Konzepte beschrieben. Im Vergleich zu anderen SNMP-MIBs, sind hier die Überwachungsaspekte am weitesten entwickelt.

2.4.1 Die Remote Monitoring Management Information Base

Die RMON-MIB erlaubt die Modellierung einer großen Bandbreite von Netzmonitoren. Die Funktionen reichen von einfachen, aktuellen Netzstatistiken, die abgefragt werden müssen, über vielfältige Alarmmöglichkeiten und Ereignismeldungen bis zu der Definition von Paketfiltern, mit deren Hilfe Paketaufzeichnungen stattfinden können. Prinzipiell wird davon ausgegangen, daß Netzmonitore, welche die RMON-MIB unterstützen, alle Meßgrößen gleichzeitig erfassen können, ohne bei hoher Netzlast oder intensiven SNMP-Abfragen an Leistungsgrenzen zu stoßen. Das Starten neuer Messungen darf unter keinen Umständen zu einer Beeinträchtigung bestehender Aktivitäten führen. Ist dies zu befürchten, muß die neue Messung vom SNMP-Agent abgelehnt werden.

In der RMON-MIB sind Objekte und Tabellen für verschiedene Meßfunktionen in Objektgruppen gegliedert, die alle optional sind. Dies gibt dem Hersteller die Freiheit, bestimmte Funktionen zu implementieren oder auch nicht. Werden einzelne Funktionen aus einem Bereich jedoch angeboten, so muß die gesamte Objektgruppe unterstützt werden. Der Hersteller besitzt jedoch keine Möglichkeit, innerhalb der RMON-MIB zusätzliche Funktionen anzubieten. Hierzu muß er auf herstellerspezifische MIB-Bereiche ausweichen.

2.4.1.1 Unterstützte Arbeitsweisen der Netzmonitore

Die RMON-MIB berücksichtigt fünf generelle Arbeitsweisen der Netzmonitore:

- Arbeiten ohne ständigen Kontakt zum Überwachungssystem (Off-line Operation),
- Erfassung der Vorgeschichte zu einem Ereignis (Pre-emptive Monitoring),
- Fehlererkennung und Übermittlung (Problem Detection and Reporting),
- Datenvorverarbeitung (Value Added Data),
- Unterstützung von mehreren Managerinstanzen (Multiple Managers).

Besitzt ein Netzmonitor einen eigenen Speicher für Meßdaten, kann er die aktuell erfaßten Informationen mit einem Zeitstempel versehen und zwischenspeichern. Auch wenn ein Netzbereich vorübergehend vom übrigen Netz abgetrennt wird, ist so eine lückenlose Erfassung der Daten möglich (Off-line Operation). Das Überwachungssystem muß die Daten nur bei Bedarf auslesen, wodurch die Belastung des Netzes mit Managementverkehr verringert wird. Der lokale Speicher ist in der Regel als Ringpuffer organisiert, damit immer die neuesten Netzgrößen zur Verfügung stehen. Die Länge des Ringpuffers ist konfigurierbar, so daß der insgesamt zur Verfügung stehende Speicher zwischen mehreren parallelen Aufzeichnungsvorgängen sinnvoll aufgeteilt werden kann. Läßt man wichtige Aufzeichnungen ständig im Hintergrund laufen, kann beim plötzlichen Auftreten von Fehlern deren Vorgeschichte für die Diagnose herangezogen werden (Pre-emptive Monitoring). Die selbständige Fehlererkennung wird durch ständigen Vergleich von standardmäßig erfaßten Größen mit konfigurierbaren Schwellen durchgeführt. Über- bzw. Unterschreitungen der Schwellwerte führen zu Ereignismeldungen (siehe Kapitel 2.4.2 und 2.4.3).

Im allgemeinen muß davon ausgegangen werden, daß ein Netz nicht nur von einem einzelnen Manager oder Überwachungssystem betreut wird. Gründe für den Einsatz mehrerer Systeme sind die Erhöhung der Ausfallsicherheit, Lastteilung, Funktions- oder Gebietsteilung. Ein Manager muß erkennen können, von welcher Managementinstanz bestimmte Konfigurationen durchgeführt wurden. Wenn mehrere Systeme auf eine MIB zugreifen, ergeben sich die typischen Datenkonsistenzprobleme. Verschärfend kommt hinzu, daß SNMP ein verbindungsloses Protokoll ist und erfordert, daß alle Aktionen idempotent, also gefahrlos mehrfach durchführbar sind. Auch können Operationen, die auf mehrere SNMP-Requests verteilt sind, nicht zu einer atomaren Einheit verbunden werden. Diese Probleme müssen durch spezielle Zugriffsmechanismen auf die Objekte der RMON-MIB gelöst werden, welche Gegenstand des nächsten Unterkapitels sind.

2.4.1.2 Verwaltung der Überwachungsaktivitäten

Der einzige Mechanismus bei SNMP, mit dem in einer MIB Objekte dynamisch erzeugt werden können, sind Tabellen. Die Struktur der Tabellen ist statisch. Über SNMP-SET-

Operationen können lediglich neue Tabellenzeilen hinzugefügt oder nicht mehr benötigte Zeilen gelöscht werden. Der SNMP-Agent ist natürlich auch in der Lage, lokal Änderungen an den Tabellen durchzuführen. In der RMON-MIB werden die Überwachungsaktivitäten eines Netzmonitors in sogenannten Steuertabellen (*Control Tables*) beschrieben und können durch Setzen der Tabelleninhalte konfiguriert werden. Typischerweise bezeichnet eine Zeile in einer Steuertabelle einen Überwachungsvorgang. Einträge können vom SNMP-Agent selbst herrühren, oder der Manager kann durch Hinzufügen einer neuen Zeile selbst eine neue Aktivität starten. Jedem Eintrag in der Steuertabelle entsprechen mehrere Einträge in korrespondierenden Datentabellen (*Data Tables*), welche die Ergebnisse des Überwachungsvorganges enthalten und über SNMP nur zum Lesen freigegeben sind. Sie enthalten eine Referenz auf den zugehörigen Steuertabelleneintrag, nach der sie auch primär indiziert sind, d.h. die Referenz steht im Namen des Objektes gleich nach der Klassenbezeichnung (erster Teil des Suffix). Damit ist es möglich, die Daten von verschiedenen Überwachungsaktivitäten in einer Tabelle abzulegen und trotzdem auf die Werte einer Aktivität gezielt zuzugreifen. Wird ein neuer Eintrag in die Steuertabelle hinzugefügt, muß der Agent, der für die Konsistenz zwischen Steuer- und Datentabelle zuständig ist, auch die entsprechenden Einträge in den Datentabellen erzeugen. Dasselbe gilt für das Löschen von Einträgen aus der Steuertabelle.

Bei der Unterstützung von mehreren Managern besteht das Hauptproblem in der konfliktfreien Erzeugung eines Eintrages in einer Steuertabelle. Es muß verhindert werden, daß, wenn zwei Manager versuchen, denselben Eintrag zu erzeugen, aufgrund des verbindungslosen Kommunikationsprotokolls ein inkonsistenter Eintrag entsteht. Deshalb wird das Anlegen eines neuen Tabelleneintrages über ein Statusobjekt vom Typ EntryStatus gesteuert. Dieses Objekt wird vom Manager und vom Agent gemäß dem Zustandsmodell nach Bild 2.10 gesetzt.

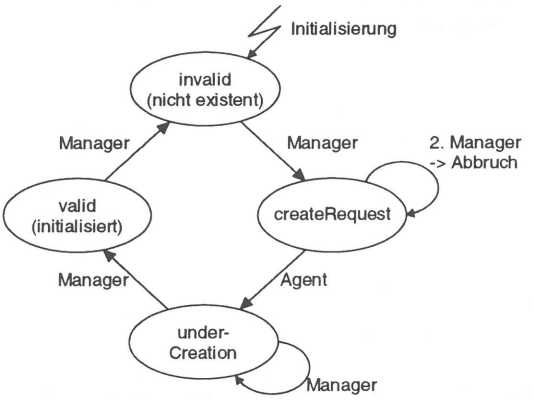


Bild 2.10: Zustandsmodell eines Steuertabelleneintrages der RMON-MIB

Bei leeren bzw. nicht vorhandenen Tabelleneinträgen ist der Wert des Statusobjektes invalid. Wird dieser Wert vom Manager auf createRequest gesetzt, legt der Agent den Eintrag intern an und setzt das Objekt selbst auf den Wert underCreation. Erst danach quittiert der Agent den SNMP-SET-Request positiv. Versucht ein anderer Manager während dieser Zeit ebenfalls diesen Eintrag anzulegen, wird dessen Request negativ quittiert. Im Zustand underCreation kann der Agent sicherstellen, daß nur der Manager, der den Eintrag erzeugte, die restliche Initialisierung des Tabelleneintrages vornehmen kann. Die Initialisierung ist abgeschlossen, wenn der Manager den Zustand auf valid setzt. Ab diesem Zeitpunkt kann der Tabelleneintrag nicht mehr verändert werden, außer ein beliebiger Manager setzt den Wert des Statusobjektes auf invalid, wodurch der Eintrag wieder gelöscht wird. Sollen die Parameter eines Überwachungsvorganges modifiziert werden, ist der zugehörige Steuertabelleneintrag zuerst zu löschen und danach neu zu erzeugen.

Die Unterstützung des Zugriffs durch verschiedene Manager erfordert neben dem gegenseitigen Ausschluß konkurrierender Schreibzugriffe auch die kooperative Nutzung der zur Verfügung stehenden Betriebsmittel. Es ist deshalb wichtig, daß Überwachungsaktivitäten den Managern zugeordnet werden können. Dies erfolgt über eine Zeichenkette (OwnerString), die in jedem Steuertabelleneintrag enthalten ist und so gesetzt wird, daß der Urheber des Eintrages identifiziert werden kann.

2.4.1.3 Objektgruppen

Die in der RMON-MIB vorhandenen Informationen beschränken sich auf Netze nach dem ISO-Standard 8802/3 (Ethernet). Erweiterungen für Netze mit anderen Medienzugriffsverfahren, wie Token-Ring oder FDDI, sind jedoch vorgesehen. Die RMON-MIB ist in neun Objektgruppen gegliedert. Sie lauten:

- Statistics,
- History,
- Hosts,
- HostTopN,
- Matrix,
- Event,
- Alarm,
- Packet Filter, und
- Packet capture.

Die Gruppen Statistics, Hosts und Matrix erfassen Zählgrößen, wie die Anzahl von Paketen oder Bytes. Für jede dieser Gruppen ist standardmäßig ein Überwachungsvorgang für jeden

Netzzugang eingerichtet. Die Gruppen History und HostTopN basieren auf einer Datenerfassung, die vom Manager selbst zu konfigurieren ist.

Die Gruppe *Statistics* besteht lediglich aus einer Tabelle, deren Zähler sich auf alle Pakete des Netzsegmentes beziehen und den momentanen Gesamtverkehr recht genau charakterisieren. Für jeden Netzzugang existiert eine Tabellenzeile. Diese Gruppe besitzt als einzige keine Steuertabelle, da jede Zeile einer Überwachungsaktivität entspricht. Erfaßt werden:

- die Netzauslastung in Prozent,
- die Anzahl aller Pakete,
- die Anzahl aller Bytes,
- die Anzahl der Broadcast-Pakete,
- die Anzahl der Multicast-Pakete,
- die Anzahl der fehlerhaften Pakete, nach Fehlerursachen aufgeschlüsselt und
- die Anzahl der Pakete, aufgeschlüsselt nach der Paketlänge.

In der *History*-Gruppe werden die wichtigsten Werte der Gruppe Statistics periodisch abgetastet und in einer Tabelle gespeichert, um später vom Manager ausgelesen werden zu können. Es können mehrere Überwachungsaktivitäten konfiguriert werden, welche die abgetasteten Werte in jeweils logisch getrennten Ringpuffern speichern. Das Abtastintervall und die Anzahl der Puffereinträge ist konfigurierbar. Dadurch ist es möglich, Aufzeichnungen in verschiedenen Zeitmaßstäben parallel zueinander durchzuführen.

In der Gruppe *Hosts* sind die Zählerwerte der Netzstatistik nach Stationen aufgeschlüsselt. Jede auf dem Netz aktive Station belegt einen Eintrag in einer Host-Tabelle (*hostTable*). Diese Tabelle ist nach den Schicht-2-Adressen der Stationen indiziert, damit der Eintrag zu einer Station direkt angesprochen werden kann. Einer Überwachungsaktivität wird eine konfigurierbare maximale Anzahl von Einträgen in der Host-Tabelle zugeordnet. Wird eine noch nicht erfaßte Schicht-2-Adresse entdeckt, führt dies zu einer neuen Tabellenzeile. Beim Überschreiten der maximalen Anzahl werden alte Einträge gelöscht. Für jede Station werden die gesendeten bzw. empfangenen Pakete und Bytes, sowie die gesendeten Broadcast-, Multicast- und fehlerhaften Pakete gezählt.

Die *HostTopN*-Gruppe setzt die Unterstützung der Hosts-Gruppe voraus. Eine Überwachungsaktivität stellt eine einmalige Messung dar, welche als Ergebnis eine Stationsliste hat, die nach den Werten eines Zählerobjektes aus der Host-Tabelle (z.B. der Anzahl der gesendeten Broadcast-Pakete) sortiert ist. Die Parameter der Messung sind das zu betrachtende Zählerobjekt, die Länge der Liste, der Startzeitpunkt der Messung, die Meßdauer und eine Referenz auf die zugehörige Überwachungsaktivität der Hosts-Gruppe. Die Stationsliste enthält die Schicht-2-Adressen der Stationen und die Differenzen der Zählerstände vor und nach

der Messung. Die Messung kann durch Setzen der Meßdauer auf einen Wert größer Null einfach wiederholt werden, wobei eine eventuell noch laufende Messung abgebrochen wird.

Detaillierte Informationen über die Verkehrsbeziehungen zwischen Stationen finden sich in der *Matrix*-Gruppe. Sie enthält eine i.A. unvollständige Verkehrsmatrix der Stationen an einem Netzsegment basierend auf den Schicht-2-Adressen. Die Anzahl der Pakete, Bytes und Fehler zwischen jeweils zwei Stationen werden, analog der Host-Statistik, in zwei Tabellen dargestellt. Die erste Tabelle ist nach Sende- und Empfangsadresse indiziert, die zweite nach Empfangs- und Sendeadresse. Beide Tabellen liefern identische Informationen, wenn sie komplett ausgelesen werden. Die Indizierung erlaubt jedoch Fragen der Form „An welche Stationen sendet Station X ?" bzw. „Von welchen Stationen empfängt Station Y?" leicht zu beantworten. Die maximale Anzahl von Tabelleneinträgen für eine Überwachungsaktivität ist auch hier begrenzt, was dazu führen kann, daß die Statistik über Verkehrsbeziehungen, die längere Zeit inaktiv sind, gelöscht werden.

Die absoluten Zählerstände der oben beschriebenen Gruppen besitzen wenig Aussagekraft, da sie in keinem Zeitbezug stehen und überlaufen können. Der Manager muß deshalb durch Polling aus diesen Zählergrößen zeitbezogene Raten berechnen. Aufgrund der Dynamik der Einträge in den Datentabellen der Hosts- und Matrix-Gruppen sind die dortigen Werte nicht miteinander vergleichbar, da nicht bekannt ist, wie lange der Eintrag schon existiert. Die Berechnung der Raten wird erschwert, da neue Einträge entstehen und alte verschwinden können.

Über die Gruppen *Alarm* und *Event* können in einem Agent allgemeine Überwachungsaktivitäten definiert und spezifische Ereignismeldungen an den Manager übermittelt werden. Sie werden in den Unterkapiteln 2.4.2 und 2.4.3 genauer erläutert.

Die Gruppen *Packet Filter* und *Packet Capture* ermöglichen einem Manager die Verwendung des Netzmonitors als eine entfernte Station zum Aufzeichnen von bestimmten Paketen auf dem Netz. Hierzu wird in der Steuertabelle der Gruppe Packet Filter ein Kanal (*channel*) definiert. Ein Kanal hat die Eigenschaft, nur Pakete von einer bestimmten Netzanschaltung, die bestimmten Filterbedingungen entsprechen, durchzulassen. Das Konzept, wie ein Filter definiert wird, ist in Kapitel 2.4.4 beschrieben. Durchgelassene Pakete werden überdies gezählt und können Ereignisse auslösen. Der Zustand eines Kanals (aktiv/inaktiv) kann sowohl durch eine SNMP-SET-Operation, als auch durch dem Kanal zugeordnete Ereignisse gesteuert werden. So können z.B. Alarme als Triggerbedingungen für den Start einer ansonsten ruhenden Aufzeichnung konfiguriert werden. Die durchgelassenen Pakete stehen anderen Objektgruppen, wie z.B. der Gruppe Packet Capture zur Verfügung. Diese Gruppe erlaubt die Speicherung von ganzen Paketen oder Paketköpfen und damit die Aufzeichnung von Vorgängen auf dem Netz. Eine Überwachungsaktivität entspricht der Speicherung aller Pakete, die den zugeordneten Kanal passieren. Der Manager kann die gespeicherten Pakete über SNMP auslesen und auswerten. Ist der Datenspeicher im Netzmonitor voll, wird die Messung entweder beendet, oder es werden

für Dauermessungen alte Einträge überschrieben. Es ist vorgesehen, daß mehrere Paketströme mit unterschiedlichen Filtern parallel aufgezeichnet werden.

2.4.2 Der Alarm-Mechanismus

Alarm ist eine allgemein verwendbare Gruppe, die es ermöglicht, daß ein SNMP-Agent selbständig Überwachungsaufgaben durchführt und nur beim Auftreten einer Fehlersituation den Manager durch eine Meldung informiert. In einer Alarmtabelle können mehrere Überwachungsvorgänge konfiguriert werden. Für jeden Vorgang werden die Werte beliebig angebbarer SNMP-Objekte vom Typ Integer periodisch abgefragt und mit einer Schwelle verglichen. Zur Beobachtung von Zählerobjekten basiert der Vergleich auf der Differenz zwischen zwei aufeinanderfolgenden Abtastwerten. Problematisch bei dieser Differenzbildung über die Abtastperiode ist, daß Spitzenwerte der Rate an den Intervallgrenzen nicht erkannt werden, obwohl sie in der Mitte eines Abtastintervalls zu einer Überschreitung des Schwellwertes führen würden. Deshalb empfiehlt der Standard, die Abtastung mit der doppelten Rate durchzuführen und einen gleitenden Mittelwert über zwei Abtastintervalle zu berechnen. Beim Über- bzw. Unterschreiten der Schwelle werden Ereignismeldungen generiert. Um die Meldungshäufigkeit in Grenzen zu halten, besitzt die Schwelle eine Hysterese, die, wie in Bild 2.11 dargestellt, über die Angabe eines oberen und eines unteren Schwellwertes eingestellt werden kann.

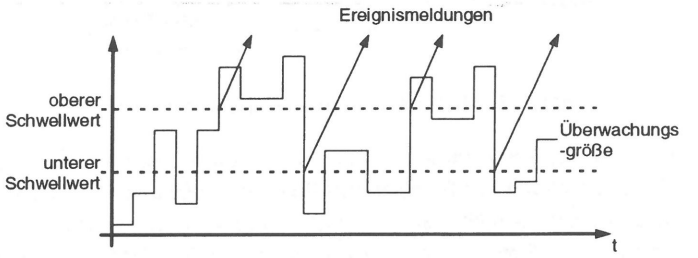


Bild 2.11: Schrankenüberwachung bei der Alarm-Gruppe

Der Alarm-Mechanismus wurde in dieser Form erstmals in der Alarm-Gruppe der RMON-MIB definiert. In der Nachfolgeversion von SNMP, dem SNMPv2, ist dieser Teil auch in die Manager-to-Manager-MIB [16] fast unverändert aufgenommen worden.

2.4.3 Ereignismeldungen

Die Gruppe *Event* ermöglicht die Konfiguration der Behandlung von Ereignismeldungen, die aus beliebigen Bereichen der MIB herrühren können. Mögliche Ereignismeldungen sind in einer globalen Event-Tabelle enthalten. Jeder Eintrag umfaßt folgende Elemente:

- eine Zeichenkette, die das Event beschreibt,
- eine Angabe, ob das Event dem Manager über einen SNMP-Trap mitgeteilt werden soll,
- eine Angabe, ob das Event an eine Log-Tabelle angehängt werden soll,
- den Community-String des zu versendenden Traps und
- den Zeitpunkt des letzten eingetretenen Ereignisses.

Bild 2.12 zeigt ein Modell der Ereignisverwaltung bei SNMP. Der auslösenden Bedingung für eine Ereignismeldung ist ein SNMP-Objekt zugeordnet, welches einen Index in die Event-Tabelle darstellt. Tritt die Bedingung ein, hängt das weitere Vorgehen des Agents von dem Inhalt des Eintrages in der Event-Tabelle ab. Dadurch kann der Manager steuern, welche Meldungen gespeichert und welche als SNMP-Trap verschickt werden sollen. Die Speicherung von Ereignissen erfolgt in einer Log-Tabelle, die ebenfalls zu der Event-Gruppe gehört. Neben der Erzeugung eines Log-Eintrages oder des Versendens eines Traps können in anderen Stellen der MIB Aktionen definiert sein, die beim Auftreten einer bestimmten Ereignismeldung ausgeführt werden. So kann z.B. auf diese Weise ein Kanal für die Aufzeichnung von Netzpaketen aktiviert werden.

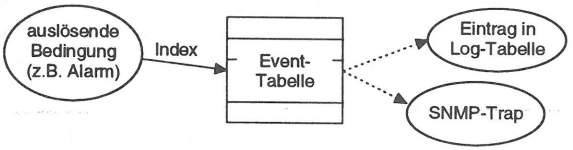


Bild 2.12: Modell der Ereignisverwaltung bei SNMP

In der RMON-MIB sind Ereignismeldungen nur bei den Gruppen Alarm und Packet Capture definiert. Weder in der RMON-MIB noch in der MIB-II sind Objekte definiert, die angeben, an welche Managerstationen SNMP-Traps verschickt werden. Dies muß durch eine Konfigurierung vor Ort oder durch Eintragungen in herstellerspezifische Teile der MIB erfolgen. Es gibt auch keine Möglichkeit über die Event-Tabelle die auslösenden Bedingungen oder die an ein Event gekoppelten Aktionen zu bestimmen. Es ist lediglich der umgekehrte Weg vorhanden.

2.4.4 Paketfilter

Paketfilter repräsentieren Bedingungen, die ein Paket erfüllen muß, um einen bestimmten Kanal passieren zu können. Diese Bedingungen werden als Einträge in einer globalen Filtertabelle gespeichert und beziehen sich auf den Inhalt und den Empfangsstatus eines Paketes. Im Empfangsstatus sind die verschiedenen Empfangsfehler in einzelnen Bits codiert. Ein Paketfilter vergleicht sowohl den Empfangsstatus, als auch den Paketinhalt mit jeweils einem Bitmuster. Dieser Vergleich kann durch zwei Masken beeinflußt werden. Die erste Maske markiert die Bits, die vom Vergleich ausgeschlossen werden sollen und die zweite Maske ändert den Ver-

gleich einzelner Bitstellen von Gleichheit auf Ungleichheit. Die Filterung nach dem Paketinhalt ist nicht nur auf den Anfang des Paketes beschränkt, sondern beginnt ab einer durch einen Offset-Wert festgelegten Position. Ein Paketfilter ist genau einem Kanal zugeordnet, wobei ein Kanal mehrere Filter besitzen kann. Damit ein Paket einen Kanal passieren kann, muß es mindestens einem Filter, der eine Referenz auf den Kanal besitzt, entsprechen. Über die Verknüpfung der Paketfilter kann, analog zur disjunktiven Normalform, eine beliebige Durchlaßbedingung für ein Paket definiert werden.

2.4.5 Abschließende Betrachtungen

Das Adressierungsschema von SNMP bedingt, daß Objektklassen, die in mehreren MIB-Gruppen benötigt werden, mehrfach definiert werden müssen. So wird z.B. die Netzstatistik in den Gruppen Statistics und History unabhängig voneinander definiert. Es kann durchaus vorkommen, daß in einem System dieselbe Managementinformation an verschiedenen Stellen in der MIB zu finden ist.

Problematisch ist auch, daß N:1-Relationen zwischen zwei Tabellen nur in einer Richtung existieren. Damit N Elemente aus Tabelle A einen Bezug auf ein Element der Tabelle B besitzen, muß in Tabelle A eine Referenz auf Tabelle B eingetragen werden. Das bedeutet jedoch, daß für die Rückwärtsbeziehung die ganze Tabelle A ausgelesen werden muß, damit diese Beziehung hergestellt werden kann. Bei komplexeren MIBs entstehen so schnell Beziehungen zwischen Objekten, die an die Managerapplikation erhöhte Anforderungen stellen. Ein Beispiel sind die Kanäle zur Paketaufzeichnung mit den dazugehörigen Filtern. Da die Filter Referenzen auf den Kanal besitzen, müssen diese beim Löschen eines Kanals mit gelöscht werden, damit bei der Wiederverwendung des Eintrags in der Kanaltabelle die Filter nicht unbeabsichtigterweise wieder wirksam werden. Ebenso komplizierte Auswirkungen hat das Löschen von Einträgen in der Event-Tabelle.

Die vollständige Implementierung der RMON-MIB in einem Netzmonitor erfordert sehr leistungsfähige Hardwarekomponenten. Vor allem die vielfältigen Konfigurationsmöglichkeiten machen es schwierig, die dazu erforderliche Rechenleistung abzuschätzen und aufgrund von Engpässen neue Überwachungsaktivitäten abzulehnen. Der Bearbeitungsaufwand, der für die Paketstatistik, Filterung und Paketaufzeichnung erforderlich ist, kann bei hohen Paketraten auf dem Netz schnell die Leistungsgrenzen überschreiten, da die Auswertungen in Echtzeit durchgeführt werden müssen.

Die Zusammenarbeit von mehreren Managern bei SNMP beruht auf einer hohen Selbstdisziplin. Die MIB bietet wenig Sicherheit vor der Manipulation einer Überwachungsaktivität eines Managers durch einen anderen zugelassenen Manager. Es existiert auch keine Trennung zwischen einer Managerstation, den Anwendungsprogrammen und den Personen, die diese Programme benutzen. Der Community-String ist hierfür kein adäquates Sicherheitsmittel.

Trotz allen Schwachpunkten bestechen SNMP und die RMON-MIB durch ihre Einfachheit bezüglich der Implementierung einer Agentapplikation. Schon früh nach dem Erscheinen der entsprechenden RFCs waren SNMP-Implementierungen auch im Quellcode frei verfügbar und lieferten die Vorlage für die Hersteller von LAN-Netzkomponenten, die heutzutage fast alle mit einem SNMP-Agent ausgestattet sind. Zwar können mit den einfachen MIB-Strukturen von SNMP auch komplexere Überwachungs- und Managementmechanismen realisiert werden, wie die Alarm- Event- und Filter-Gruppen zeigen, der Kommunikationsaufwand ist jedoch relativ hoch, da häufig ganze Tabellen ausgelesen und Netzkomponenten gepollt werden müssen. Dies führt bei zentraler Überwachung größerer Netze zu einer erheblichen Netzbelastung durch den Managementverkehr. Trotzdem kann es sinnvoll sein, die Komplexität der Netzüberwachung in eine zentrale Managerstation zu verlagern, um die verteilten Komponenten einfach zu halten.

2.5 Funktionen für die Netzüberwachung im OSI-Netzmanagement

Die in Kapitel 2.4 vorgestellte RMON-MIB ist auf die spezielle Meßsystemklasse der Netzmonitore zugeschnitten, deren Aufgabe die Beobachtung des Verkehrs auf einem Netzsegment ist. Damit auch andere Meßsysteme möglichst einheitlich verwendbar sind, müssen deren Funktionen standardisiert werden, ohne jedoch die Systeme selbst zu nivellieren. Dies wird dadurch erreicht, daß die Meßfunktionen in allgemeine Grundfunktionen strukturiert werden, die dann in Managed Objects und Operationen auf diese Objekte modelliert werden können. Die OSI Systems Management Functions [63] definieren solche generischen Funktionsblöcke, die in verschiedenen Funktionsbereichen Verwendung finden können. Die für die Netzüberwachung wesentlichen Funktionen sind:

- die Übermittlung von Ereignismeldungen und Alarmen,
- die Beobachtung wichtiger Betriebsparameter und
- die Durchführung von Tests im Netz.

Nachfolgend werden diese, für die Netzüberwachung wichtigen, standardisierten Managementfunktionen erläutert.

2.5.1 Ereignismeldungen und Alarme

Über Ereignismeldungen kann ein Agent einen Manager über spontane Veränderungen im Netz unterrichten. Im Gegensatz zu SNMP-Traps sind diese beim OSI-Management Bestandteil der Managed Objects und enthalten wesentlich spezifischere und detailliertere Informationen. Die Ereignismeldung eines Managed Objects wird als Notifikation bezeichnet. Die Übertragung dieser Notifikation an einen Manager oder ein anderes Managed Object erfolgt über einen Dienst, der spezifisch für den Typ der Notifikation ist, letztendlich jedoch auf den CMIS-

Dienst M-EVENT-REPORT aufsetzt. Der spezifische Dienst definiert lediglich zusätzliche Parameter innerhalb des CMIS-Event Reports, welche die Informationen der Notifikation enthalten. Werden große Netze überwacht, ist es wichtig, daß ein Manager kontrollieren kann, welche Event Reports an ihn geschickt werden, um zu verhindern, daß das Netz und der Manager durch viele redundante oder nicht relevante Meldungen belastet wird.

Der OSI-Standard 10164-5, *Event Reporting Function* [65] definiert ein Konzept, wie Notifikationen selektiv an verschiedene Manager übertragen werden können. Das zugrunde liegende Modell zeigt Bild 2.13. In der Event-Vorverarbeitung wird aus der Notifikation eines Managed Objects die Datenstruktur für einen Event Report erzeugt und ausgefüllt. Die Auswahl, an welche Manager dieser Event Report verschickt wird, treffen Managed Objects der Klasse *Event Forwarding Descriptor* (EFD). Jeder Manager legt für jede Ereignismeldung, welche er von einem Agent erhalten möchte, ein EFD-Objekt an. Bei mehreren Managern kann dies zu einer erheblichen Anzahl führen, die alle von dem Agent verwaltet werden müssen.

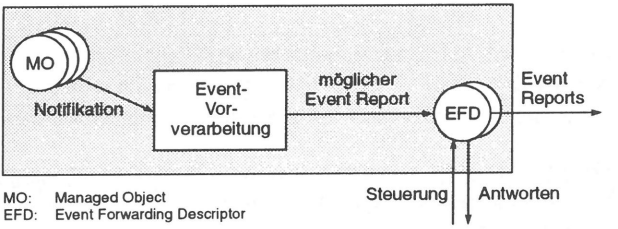


Bild 2.13: Modell der Event Reporting Function

Ein EFD-Objekt enthält die Adresse der Anwendungsinstanz eines Managers und die Selektionskriterien, anhand derer entschieden werden kann, ob die Meldung tatsächlich übermittelt werden soll. Ist ein Manager vorübergehend nicht erreichbar, kann der Event Report an andere Manager (*Backup Destinations*) geschickt werden, damit er nicht verloren geht. Ein optionales *Scheduling Package* legt fest, für welche Tageszeiten oder Wochentage das EFD-Objekt aktiv ist. Ein Manager kann seine EFD-Objekte kontrollieren, die Selektionskriterien auslesen und modifizieren.

Alarme sind besondere Notifikationen für die Unterrichtung über Alarmzustände innerhalb eines Managed Objects. Die *Alarm Reporting Function* [64] stellt allgemeine Notifikationen für die Übermittlung von Fehlerzuständen (*Trouble Tickets*) zur Verfügung und definiert den Dienst *Alarm Reporting Service* zur Übermittlung dieser Alarme in Event Reports. Sowohl das Eintreten als auch die Beendigung eines Alarmzustandes wird angezeigt. Alarme umfassen fünf Event Report-Typen, *Communications Alarm, Quality of Service Alarm (QOS Alarm), Processing Error Alarm, Equipment Alarm*, und *Environmental Alarm*. Zusätzliche Parameter des Event Reports sind der Schweregrad des Alarms und eine mögliche Alarmursache. Es wurden

viele mögliche Ursachen standardisiert, wie z.B. *IO Device Error, LAN Error* oder *Out of Memory*. Untereinander korrelierte Alarmmeldungen enthalten jeweils eine Liste mit Referenzen auf die anderen Alarme.

Die *Security Alarm Reporting Function* [67] beschreibt Alarme, die von der Überwachung sicherheitsrelevanter Objekte herrühren. Sie definiert eine Notifikation zur Signalisierung von Fehlfunktionen der Sicherheitsmechanismen bzw. von erfolgreichen oder erfolglosen Sicherheitsverstößen. Die Sicherheitsalarme sind unabhängig von den Alarmen der Alarm Reporting Function. Die hier definierten Event Report-Typen sind:

- Integrity Violation (Datenintegritätsverletzungen),
- Operational Violation (Fehler bei der Erbringung des Dienstes),
- Physical Violation (Einbruch in Systeme),
- Security Service Mechanism Violation (Sicherheitsattacke),
- Time Domain Violation (Sicherheitszeitschranke abgelaufen).

Weitere Parameter sind die Alarmursache, der Schweregrad und optional ein freier Text, mit dem das Problem beschrieben werden kann. Zusätzlich können noch weitere Daten angegeben werden, deren Struktur in diesem Zusammenhang jedoch nicht weiter definiert ist.

Grundsätzlich können alle Notifikationen eines Managed Objects auch im Agent lokal, in *Logs* aufgezeichnet und somit längerfristig gespeichert werden. Die *Log Control Function* [66] definiert dafür ein der Event Reporting Function sehr ähnliches Modell. Ein Log ist ein Managed Object, welches mehrere Objekte der Klasse *logRecord* enthalten kann. Damit die Informationen, die mit einem Event Report verschickt werden, auch in einen Log gespeichert werden können, ist von der Objektklasse logRecord die Klasse *eventLogRecord* abgeleitet, welche die zusätzlichen Ereignisparameter beinhaltet. Zur Speicherung spezieller Notifikationen, wie z.B. Alarme oder Sicherheitsalarme, werden wiederum spezielle Record-Klassen abgeleitet (hier: *alarmRecord* und *securityAlarmReportRecord*).

2.5.2 Beobachtung der Netzbetriebsdaten

Im OSI-Netzmanagement sind die Erfassung der Daten über das Netz bzw. deren Systeme von den eigentlichen Überwachungsfunktionen, d.h. dem Sammeln, Vorverarbeiten und Übertragen von Zustandsinformationen an einen zentralen Manager, getrennt. Dies erlaubt, Überwachungsfunktionen in Support Managed Object-Klassen zu modellieren, die völlig unabhängig von den zu überwachenden Meßgrößen sind. Die Überwachungsaktivitäten sind direkt an die Instanzen dieser Klassen gekoppelt und können vom Manager erzeugt und gelöscht werden. Um ein Polling durch den Manager möglichst zu vermeiden, wurden Funktionen definiert, die es erlauben, auf verschiedenen Ebenen ein Managed Object zu beobachten. Dabei wurden auch bestimmte Verfahren der Datenvorverarbeitung standardisiert. Die *Summarization Function*

[70] standardisiert die grundlegende Funktionalität, ein oder mehrere Managed Objects zu beobachten, d.h. bestimmte Attribute zyklisch auszulesen, zusammenzufassen und das Ergebnis einem Manager über Ereignismeldungen mitzuteilen. Die Überwachung von Leistungskenngrößen eines Netzes ist Gegenstand der *Workload Monitoring Function* (WMF) [68]. Sie standardisiert Metriken, welche die Leistung eines Netzbetriebsmittels charakterisieren, und beschreibt ein Modell zur Überwachung dieser Metriken durch Vergleich mit Schwellen. Darüber hinaus werden Methoden zur statistischen Vorverarbeitung des zeitlichen Verlaufs der Leistungsgrößen definiert.

2.5.2.1 Zyklische Abfrage von aktuellen Werten aus Managed Objects

Die Summarization Function definiert mit der Objektklasse *Scanner* ein dynamisch erzeugbares Objekt, welches in der Lage ist, mehrere Attribute anderer Managed Objects zu beobachten. Ein Scanner-Objekt kann, wie in Bild 2.14 dargestellt, zu dem beobachteten Objekt in einer Enthaltenseins-Beziehung stehen oder sich auf einem anderen System befinden. Dies ist besonders bei einfachen Systemen vorteilhaft, um durch die Auslagerung des Scanner-Objektes den Agent einfach zu halten. Der Bezug zu dem beobachteten Wert entsteht dadurch, daß Objekt- und Attributname in Attributen des Scanner-Objektes abgelegt werden.

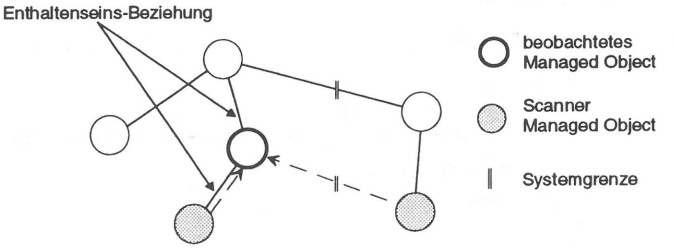


Bild 2.14: Überwachung durch Scanner-Objekte

Von der Klasse Scanner sind weitere Objektklassen abgeleitet, von denen manche die abgefragten Werte statistisch zusammenzufassen. Bei den Klassen der Summarization Function wird jedoch keine zeitbezogene Statistik berechnet. Die Objektklasse *simpleScanner* gibt als einfachste Scanner-Klasse alle Werte der ausgelesenen Attribute in einer speziellen Notifikation, dem *Scan Report*, an den Manager weiter. Die Klasse *dynamicSimpleScanner* wird erst durch eine Aktion zu einer einmaligen Messung angestoßen, in der die zu beobachtenden Objekte und Attribute dynamisch festgelegt werden. Die Klassen *meanScanner*, *meanVarianceScanner*, *minMaxScanner* u.a. fassen die ausgelesenen Attributwerte entsprechend ihrem Namen zu einer *Statistical Report*-Notifikation zusammen, verhalten sich aber sonst wie ein Simple-Scanner-Objekt. Sollen unterschiedliche Attributtypen aus verschiedenen Objekten abgefragt werden, ist die Klasse *heterogeneousScanner* zu verwenden. Schließlich kann der *Buffered*

Scanner, der wie ein Heterogeneous Scanner arbeitet, alle abgefragten Werte intern speichern, so daß sie später ausgelesen werden können.

2.5.2.2 Leistungsüberwachung von Netzbetriebsmitteln

Die Workload Monitoring Function standardisiert drei Metriken, welche die Leistung eines ganz allgemein gehaltenen Netzbetriebsmittels charakterisieren:

- die Auslastung (*resource utilisation*),
- die Rate der Dienstanforderungen (*resource request rate*) und
- die Rate der abgewiesenen Dienstanforderungen (*rejection rate*).

Die Werte dieser Metriken werden über ein konfigurierbares Zeitintervall berechnet. Für die Überwachung sind eine Reihe von unterschiedlichen Schwellen (*thresholds*) definiert, deren Klassifizierung über drei Angaben erfolgt, die einen dreidimensionalen Raum, wie er in Bild 2.15 dargestellt ist, aufspannen.

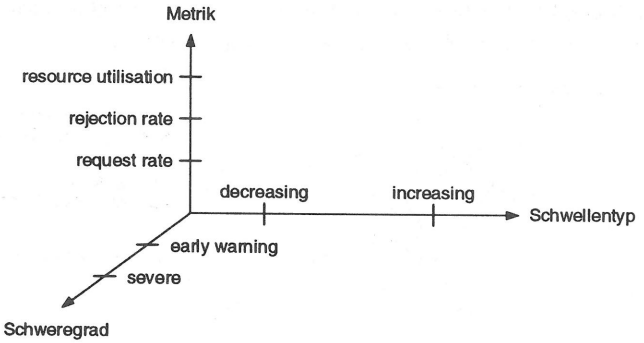


Bild 2.15: Klassifizierung der Schwellen des Workload Monitoring

Der Schwellentyp (*increasing/decreasing*) berücksichtigt die Tatsache, daß je nach überwachter Größe sowohl die Unterschreitung des Schwellwerts kritisch sein kann, wie z.B. der Anzahl freier Slots auf einem getakteten Medium, als auch die Überschreitung, wie bei der Anzahl der belegten Zeitschlitze. Der Schweregrad (*early warning/severe*) gibt einen Hinweis auf die Dringlichkeitsstufe des Problems und erlaubt abgestufte Warnungen. Es ist vorgesehen, daß diese Dimension von einem Manager um zusätzliche Stufen erweitert wird. Schließlich ist der Schwelle die überwachte Metrik zugeordnet, so daß ein Manager spezifisch auf das Überschreiten einer bestimmten Schwelle reagieren kann.

Die Überwachungsfunktionen der WMF beschränken sich auf Pegelgrößen (*gauge values*), deren Werte zwischen einer unteren und einer oberen Grenze schwanken können. Kreuzt die beobachtete Größe eine Schwelle, wird der Manager über die Notifikation *QOS Alarm Report* benachrichtigt. Ein QOS Alarm Report enthält den Attributnamen, den beobachteten Wert und den überschrittenen Schwellwert. Darüber hinaus wird dem Manager mitgeteilt, wann ein gleichartiger Alarmzustand das letzte Mal auftreten ist. Jede Schwelle besitzt zwei Schwellwerte (*Trigger Value* und *Clear Value*). Über diese Werte wird eine Hysterese realisiert, die verhindert, daß sehr viele Notifikationen erzeugt werden, falls der beobachtete Wert geringfügig um den Schwellwert schwankt. Wie die Schwellwerte zu setzen sind, ergibt sich aus dem in Bild 2.16 dargestellten Modell. Für eine Größe können Schwellen mit unterschiedlichen Schweregraden (*severityIndication*) definiert werden. Es ist festgelegt, daß der Grad *severe* die höchste Bedeutung besitzt und die übrigen Stufen weniger kritische Zustände bezeichnen. Der Schwellentyp definiert die Bedeutung der oberen und unteren Schwellwerte. Bei *Increasing Thresholds* führt das Überschreiten des oberen Schwellwerts zu einem *Threshold Event* und das Unterschreiten des unteren Schwellwerts zu einem *Threshold Clear Event*. Bei *Decreasing Thresholds* ist es genau umgekehrt. Den Schwellwerten sind Flags zugeordnet, mit denen die Erzeugung der Notifikation gesteuert werden kann.

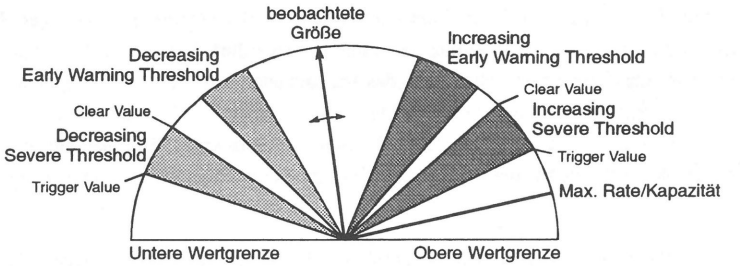


Bild 2.16: Allgemeines Schwellwertmodell einer Pegelgröße

2.5.2.3 Statistische Vorverarbeitung der Leistungskenngrößen

Überwachungsvorgänge des zeitlichen Verlaufs von Attributen werden durch sogenannte *Metric Objects* modelliert, die von der Objektklasse Scanner abgeleitet sind. Wie bei den Scanner-Objekten werden auch hier Attribute eines unter Beobachtung stehenden Managed Objects periodisch abgetastet. Da für eine zeitliche Beobachtung nur Pegelwerte in Frage kommen, werden andere Attributtypen wie z.B. Zähler in einen Ratenwert konvertiert (*Derived Gauge*). Das *Counter Difference Package* führt diese Umwandlung durch einfache Differenzbildung der Attributwerte zu Beginn und am Ende des Abtastintervalls (*Granularity Period*) durch. Auf die Pegelwerte können weitere Funktionen wie Mittelwertberechnungen oder Filteroperationen angewendet werden, bevor sie mit einer oder mehreren Schwellen verglichen werden. Die ver-

schiedenen Vorverarbeitungsfunktionen führen zu unterschiedlichen Klassen von Metric Objects. Ihre Vererbungshierarchie zeigt Bild 2.17.

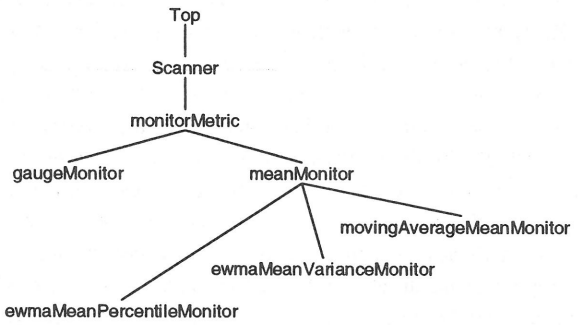


Bild 2.17: Klassenhierarchie der Metric Objects

Gemeinsame Basis ist die Klasse *monitorMetric*, welche optional das Counter Difference Package enthält. Die einfachste instanziierbare Objektklasse ist *gaugeMonitor* und vergleicht ein Attribut direkt mit einer oder mehreren Schwellen. Außer der Berechnung einer Rate für Zählerattribute wird keine Verarbeitung der Daten durchgeführt. Die Objektklasse *meanMonitor* berechnet einen gleitenden Mittelwert des beobachteten Attributs und vergleicht ihn mit dem Schwellwert. Als Parameter für die Mittelwertberechnung ist als Attribut ein Maß für die Speichertiefe enthalten. Davon abgeleitet ist der *movingAverageMeanMonitor*, eine Klasse, bei welcher der Algorithmus für die Berechnung eines gleitenden Mittelwertes angegeben werden kann.

Zwei Algorithmen stehen dafür zur Auswahl, die beide auf einem beweglichen Zeitfenster (*Sliding Time Window*) beruhen. Beim *Uniformly Weighted Moving Average*-Algorithmus (UWMA) wird das arithmetische Mittel aller Abtastwerte x(t) mit dem Abtastintervall DT innerhalb eines Zeitfensters (DT · N) gemäß Gleichung 2.1 berechnet. Hierfür ist es jedoch notwendig, daß alle N Werte innerhalb des Zeitfensters gespeichert werden.

$$\bar{x}(t) = \frac{1}{N} \cdot \sum_{i=0}^{N-1} x(t - i \cdot DT) \qquad (2.1)$$

Mit dem *Exponentially Weighted Moving Average*-Algorithmus (EWMA), der dem eines exponentiellen digitalen Filters entspricht und in den Gleichungen 2.2 und 2.3 in unterschiedlichen Schreibweisen angegeben ist, wird dies umgangen.

$$\bar{x}(t) = \bar{x}(t - DT) + (x(t) - \bar{x}(t - DT)) \cdot \frac{DT}{T_1} \qquad (2.2)$$

$$\bar{x}(t) = \frac{DT}{T_1} \cdot \sum_{i=0}^{\infty} (1 - \frac{DT}{T_1})^i \cdot x(t - i \cdot DT) \qquad (2.3)$$

Über die Zeitkonstante (T_1) wird der Einfluß des aktuellen Werts auf den Mittelwert berücksichtigt, was der Größe des Zeitfensters beim UWMA-Algorithmus entspricht. Der Mittelwert ist jedoch im Prinzip von der kompletten Vorgeschichte abhängig und aktuelle Werte werden stärker gewichtet als ältere. Dafür werden in der Formel lediglich der letzte Mittelwert und der aktuelle Abtastwert benötigt.

Die Objektklasse *ewmaMeanVarianceMonitor* berechnet neben dem Mittelwert nach dem EWMA-Algorithmus die Varianz der abgetasteten Attributwerte, vergleicht jedoch nur den Mittelwert mit dem Schwellwert. Bei Schwellwertüberschreitungen wird in der Notifikation jedoch auch die Varianz mitgeliefert. Die Klasse *ewmaMeanPercentileMonitor* unterscheidet sich von der obigen Klasse dadurch, daß sie statt der Varianz Schätzwerte für den Medianwert, die n%-Quantile, die (100-n)%-Quantile berechnet (n = [1..49]), sowie den größten und kleinsten Wert bestimmt. Überwacht wird wiederum nur der Mittelwert. Die Gleichungen für die rekursive Berechnung der Varianz und der n%-Quantile sind in [68] zu finden.

2.5.3 Tests

Die Gewinnung von Informationen über die Leistungsfähigkeit und Funktionstüchtigkeit von Betriebsmitteln des Netzes durch Tests bildet neben der passiven Beobachtung das zweite Standbein der Netzüberwachung. Allgemeine Testfunktionen werden von derzeitigen SNMP-MIBs nicht abgedeckt. Der OSI-Standard *Test Management Function* [69] befaßt sich mit Modellen, welche erlauben, Testfunktionen in Objektkatalogen auf einer abstrakten Ebene zu definieren. Es werden drei Testmethoden unterschieden:

- Loopback-Tests oder Echotests,
- Fault Injection Tests, bei denen Fehlersituationen provoziert werden und
- Selbsttests.

Tests zeichnen sich dadurch aus, daß sie in das zu überwachende System eingreifen. Ihre Durchführung kann an bestimmte Bedingungen geknüpft sein, damit z.B. verhindert wird, daß durch sie der Netzbetrieb gestört wird. Der allgemeine Testablauf beinhaltet die Schaffung einer Testumgebung, die kontrollierte Ausführung des Tests, die Wiederherstellung der normalen Umgebung nach dem Test und die Testauswertung. Die kontrollierte Testausführung beinhaltet die Möglichkeit, den Test zwischenzeitlich anzuhalten, fortzuführen oder abzubrechen.

In der Regel sind in einen Test mehrere Instanzen involviert, die sich im allgemeinen auf unterschiedlichen Systemen im Netz befinden. Die Instanz, welche den Test auslöst oder anfordert und die Ergebnisse empfängt, wird als *Test Conductor* bezeichnet. Die Testdurchführung liegt

bei der Partnerinstanz, dem *Test Performer*. Tests beinhalten u.U. die Kommunikation mit einer dritten Station, dem *Secondary Test Performer*, unter Verwendung von Standardprotokollen der zu testenden Schicht wie z.B. bei Echotests. Bild 2.18 zeigt die in einem Test involvierten Instanzen. Der Test Conductor ist innerhalb des Managers angesiedelt. Agents mit der Funktionalität eines Test Performers sind für die Durchführung von Tests verantwortlich. Der Secondary Test Performer bemerkt in der Regel nicht, daß die betreffende Kommunikation zu einem Test gehört.

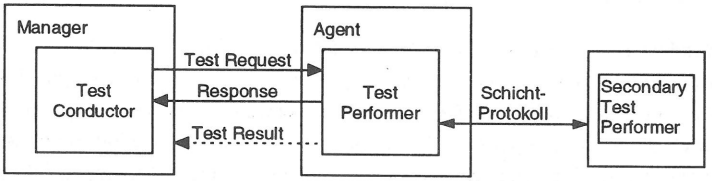


Bild 2.18: Testinstanzen

Die Funktionalität, auf eine Testanforderung (*Test Request*) zu reagieren, ist in einem Managed Object, dem *Test Action Request Receiver* (TARR) modelliert. Der Test selbst bezieht sich auf ein *Managed Object Under Test* (MOT). Für bestimmte Tests können TARR und MOT auch zusammenfallen. Es wird zwischen synchronen und asynchronen Tests unterschieden. Bei synchronen Tests muß der Manager warten, bis er die Ergebnisse erhält. Ein angestoßener Test kann nachträglich nicht mehr beeinflußt werden. Die Kommunikationsbeziehung zwischen Test Conductor und dem TARR-Objekt bleibt für die Dauer des gesamten Tests bestehen, wobei das TARR-Objekt für diese Zeit blockiert ist. Die Testergebnisse werden in einer oder mehreren Antworten auf den Testauftrag direkt an den Test Conductor übermittelt. Das Objektmodell für synchrone Tests zeigt Bild 2.19.

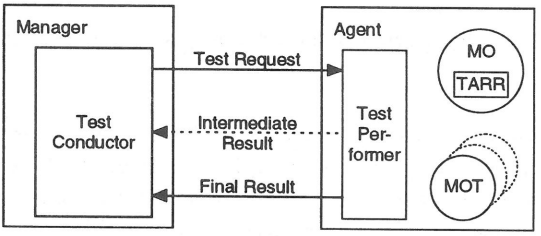


Bild 2.19: Objektmodell synchroner Test

Eine wesentliche Eigenschaft asynchroner Tests ist, daß mehrere Tests parallel ausführbar sind und daß die Testausführung über Managementoperationen gesteuert werden kann. Damit können Hintergrundtests realisiert werden. Wie in Bild 2.20 dargestellt, trifft die Testanforderung

auf ein TARR-Objekt, welches jedoch den Test nicht selbst durchführt, sondern dafür spezielle Testobjekte (*Test Objects*, TOs) erzeugt. Nach der Testbestätigung wird die Kommunikationsbeziehung zwischen dem Test Conductor und dem TARR-Objekt wieder abgebaut. Für die Adressierung der laufenden Tests erhalten die Testobjekte als Attribut eine eindeutige Kennung. Ein Test kann mehrere Testobjekte umfassen, die zueinander in Beziehung stehen, indem sie die gleiche Testkennung besitzen. Dadurch ist es möglich, komplexere Tests aus einfacheren Testobjekten zusammenzusetzen. Sowohl Testparameter als auch Testergebnisse werden durch Attribute der Testobjekte repräsentiert und können durch Managementoperationen ausgelesen bzw. modifiziert werden. Dies gibt dem Test Conductor die Möglichkeit, in den laufenden Test einzugreifen, ihn zu modifizieren und die Erzeugung von Zwischenergebnissen anzustoßen.

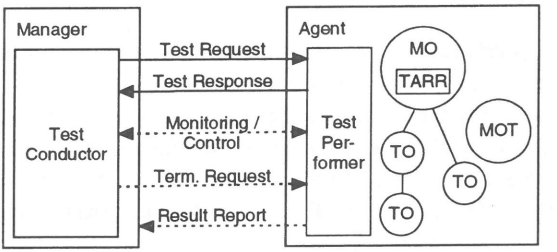


Bild 2.20: Objektmodell asynchroner Test

Testobjekte enthalten optional ein Scheduling-Package, welches erlaubt, einen Startzeitpunkt anzugeben und periodische Tests zu definieren. Die Wahl des Startzeitpunktes kann auch dem Test Performer überlassen werden, um z.B. den Test während einer Niederlastphase durchzuführen. Alle Testergebnisse werden als Notifikationen übermittelt und können somit an mehrere Manager verteilt bzw. in Log-Objekten gespeichert werden.

2.6 Überwachungssysteme

Die Netzüberwachung mit verteilten Sensoren hat letztendlich den Zweck, einem Anwender, sei es ein menschlicher Netzadministrator oder eine Netzmanagementapplikation, Informationen zu präsentieren, die bestimmte Fragestellungen beantworten oder über ungewöhnliche Vorkommnisse unterrichten. Diese Informationen sind in einer Management Information Base häufig schwer zu finden und selten in der passenden Form vorhanden. Überwachungssysteme schließen diese Lücke, indem sie die abgefragten Netzparameter und eingehenden Ereignismeldungen in Zusammenhang bringen und anwendergerecht darstellen. Bis zum gegenwärtigen Zeitpunkt stützen sich viele Überwachungskonzepte auf eine Sammlung einfacher Tools, von denen jedes eine spezielle Aufgabe erfüllt. Eine Auswahl solcher Tools für TCP/IP-Netze fin-

det sich in [132]. Weitere Tools sind die herstellerspezifischen Bedienprogramme für im Netz verteilte Meßkomponenten.

Netzmanagementplattformen ermöglichen einen einheitlichen, ortstransparenten Zugriff auf Netzmanagementinformationen. Sie integrieren unterschiedliche Kommunikationsprotokolle und ermöglichen somit die Verwaltung aller Netzkomponenten, die eine standardisierte Netzmanagementschnittstelle besitzen und damit das MIB-Konzept unterstützen. Die Open Software Foundation standardisierte mit dem *Distributed Management Environment* (DME) die Architektur dieser Plattformen [80], und erlaubt somit die eigentlichen Netzmanagementapplikationen plattformunabhängig zu erstellen und auf verschiedene Rechner im Netz zu verteilen. DME umfaßt den Informationsdienst, die Objektverwaltung und eine Schnittstelle zur grafischen Benutzeroberfläche.

Netzmanagementplattformen, die der DME-Architektur entsprechen, sind als Produkte erhältlich [38]. Ihr Schwerpunkt liegt in der Verwaltung der Netzkonfiguration, der Netztopologie und in der Anzeige und Speicherung von Ereignismeldungen. Als frei definierbare Überwachungsfunktion steht die zyklische Abfrage von MIB-Informationen mit entsprechender Darstellung zur Verfügung. Spezielle Überwachungsapplikationen existieren für die Anzeige der Werte in der RMON-MIB [39] und die Erfassung und Anzeige von Langzeitstatistiken für den stationsbezogenen Netzverkehr [41].

Kapitel 3

Ein neues Konzept für eine offene Netzüberwachung

In Kapitel 2.2 wurde die Netzüberwachung als die laufende Beobachtung des Netzes beschrieben mit dem Ziel, die Betriebsparameter aufzuzeichnen und Ausnahmesituationen zu erkennen. Sie verkörpert einen elementaren Bereich des Netzmanagements, der vergleichbar ist mit der Verwaltung der Fülle von Managementinformationen. Netzüberwachung bedeutet mehr als nur die wiederholte Ausführung bestimmter Abfragen, deren Ergebnisse angezeigt oder mit Schwellwerten verglichen werden. Sie legt vielmehr fest, wie die vielfach im Netz vorhandenen Sensoren eingesetzt werden, um den Überwachungsabsichten des Anwenders gerecht zu werden. Sie schließt dabei auch die Definition der Überwachungsaktivitäten, die Steuerung der Überwachungsabläufe und die Datenerfassung mit ein.

Die Position der Netzüberwachung innerhalb der Managementarchitektur ist nicht klar definiert. In der OSI Workload Monitoring Function sind Überwachungsfunktionen Teil der MIB und damit einem Agent zugehörig. Häufig werden sie aber auch als Komponente einer Managementplattform oder als separate Netzmanagementapplikation betrachtet. Unabhängig von der Netzmanagementarchitektur behandelt das nachfolgende Kapitel ganz allgemein die Anforderungen und Ziele, die an eine offene Netzüberwachung gestellt werden. Kapitel 3.2 stellt ein neues Konzept vor, welches alle Überwachungsaspekte abdeckt und die Realisierung eines umfassenden Überwachungssystems erlaubt. In den darauffolgenden Kapiteln wird dann auf einzelne Aspekte dieses Konzeptes eingegangen.

3.1 Anforderungen und Ziele

Netzüberwachung ist ein dynamischer Vorgang. Er muß sich stets an den Bedürfnissen des Anwenders und an den Gegebenheiten im Netz orientieren. Eine allgemeingültige Überwachungsstrategie existiert nicht. Deshalb muß der Benutzer selbst in der Lage sein, die für ihn wichtigen Überwachungsaktivitäten in einer angemessenen Form zu definieren. Die Ergebnisse

dieser Aktivitäten sind bezüglich des Detaillierungsgrades und der Art der Präsentation (graphisch oder textbasiert) anwendergerecht darzustellen. Eine offene Netzüberwachung integriert heterogene Meßsysteme mit unterschiedlichen Kommunikationsprotokollen und Steuerungsprinzipien. Sie löst damit meßkomponentenspezifische, zentrale Überwachungssysteme ab und verbirgt die Heterogenität der Meßsysteme vor dem Benutzer. Damit können von unterschiedlichen Meßsystemen erfaßte Daten gemeinsam ausgewertet und zueinander in Beziehung gesetzt werden. Die Netzüberwachung, wie sie hier betrachtet werden soll, ist gekennzeichnet durch:

- die Tatsache, daß die Informationen verteilt erfaßt werden müssen,
- eine Vielfalt von Meßsystemen, zu denen ebenfalls die Netzkomponenten, welche über das Netzmanagement abgefragt werden, zählen,
- die Unvorhersehbarkeit der eigentlichen Überwachungsaktivitäten und
- den Anspruch, alle Überwachungsaspekte des Netzmanagements abzudecken.

Aufgrund der Unbestimmtheit der Überwachungsvorgänge müssen auch neue Funktionen, die sich mit den im Netz vorhandenen Daten realisieren lassen, nachträglich in ein Überwachungssystem eingebracht werden können. Aus diesen Betrachtungen ergeben sich zusammenfassend folgende Anforderungen an eine offene Netzüberwachung:

- Durchgängigkeit des Modells für alle Überwachungsaktivitäten,
- Integration der Meßsysteme,
- Flexibilität und Erweiterbarkeit des Überwachungssystems und
- Anwenderorientierung.

Das Spektrum der Anwender ist dabei breit gefächert. Es besteht aus menschlichen Benutzern und Netzmanagementapplikationen. Ein menschlicher Benutzer in der Rolle eines Netzadministrators ist für den Betrieb des gesamten Netzes verantwortlich. Der Überwachungsschwerpunkt liegt in der Gewinnung eines Überblicks über das ganze Netz. Die Informationsdarstellung hat deshalb übersichtlich und sehr komprimiert zu erfolgen. So sollte z.B. der Status der Netzkomponenten über deren farbliche Darstellung in einer Art Netzkarte auf einen Blick erfaßbar sein.

Einen anderen Anwendertyp stellen die Personen dar, die einen bestimmten Aspekt des Netzes genauer untersuchen wollen. Sie benutzen das Überwachungssystem auch vor Ort im Rahmen einer Fehlersuche und benötigen sehr detaillierte Informationen über das Netz. Mitunter sind die graphischen Darstellungsmöglichkeiten beschränkt, so daß die Informationen in einer textbasierten Form angezeigt werden müssen. Die Überwachungsaktivitäten sind in diesem Fall überwiegend spontaner und kurzlebiger Natur, da sie begleitend zur Durchführung von Tests im Netz stattfinden.

Netzmanagementapplikationen als Anwender eines Überwachungssystems müssen in der Lage sein, Überwachungsaktivitäten zu generieren und die Ergebnisse auszuwerten. Werden diese als Managed Objects modelliert, kann die Applikation auf die Kommunikationsmechanismen des Netzmanagements zurückgreifen. Dies bedeutet jedoch sowohl auf der Applikationsseite als auch der Seite des Überwachungssystems einen entsprechenden Umsetzungsaufwand. Eine andere Gestaltungsmöglichkeit der Schnittstelle besteht darin, die Aufbereitung der Netzdaten so an die Applikation anzupassen, daß eine einfache Form des Datenaustausches, z.B. temporäre Dateien, gewählt werden kann.

Das prinzipielle Problem bei der Netzüberwachung besteht darin, daß, wie in Bild 3.1 dargestellt, aus einer großen Menge von Daten über das Netz und den darin befindlichen Systemen die wesentlichen Informationen ausgewählt und für den Anwender, hier allgemein als Netzmanagement bezeichnet, aufbereitet werden müssen. Die Informationsauswahl ergibt sich dabei aus den Überwachungsabsichten des Netzmanagements.

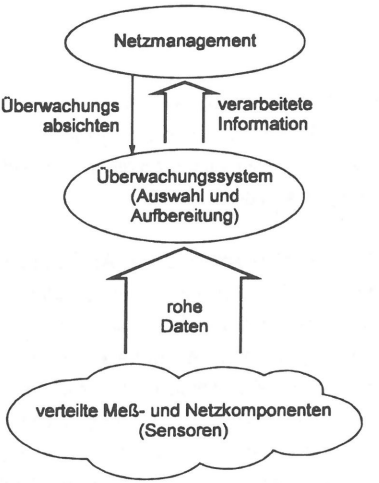


Bild 3.1: Informationsfluß bei der Netzüberwachung

Die Art der Aufbereitung ist abhängig vom Einsatzgebiet der Überwachungsaktivität. So besteht z.B. beim Fehlermanagment die Überwachung im wesentlichen aus dem Erkennen von plötzlich auftretenden Fehlersymptomen. Jedes Symptom ist an eine oder mehrere Überwachungsgrößen gekoppelt und tritt auf, wenn diese Größen einen definierten Normalbereich verlassen. In diesem Zusammenhang gestalten sich Überwachungsvorgänge als periodische Abfragen mit anschließendem Vergleich. Treten Fehler auf, werden im Rahmen der Fehlerlokalisation kurzfristig zusätzliche Netzgrößen einmalig abgefragt. Da hierbei häufig auch Infor-

mationen zur Vorgeschichte des Fehlers notwendig sind, müssen Überwachungsaktivitäten existieren, die diese Werte laufend erfassen und für eine gewisse Zeit zwischenspeichern. Beim Leistungsmanagement steht die korrekte Berechnung aussagekräftiger Kenngrößen aus den erfaßten Daten im Vordergrund. Diese Kenngrößen werden visualisiert bzw. für längerfristige Trendanalysen gespeichert. Für das Konfigurationsmanagement ist es wichtig, daß der aktuelle Status des Netzes jederzeit verfügbar ist und die Zeitpunkte der Statuswechsel als Ereignisse dem Netzmanagement mitgeteilt werden. Ermittelt wird dieser Status durch periodische Erreichbarkeitstests und Abfragen.

Zur Minimierung der Datenerfassung und der Übertragung muß der Informationsfluß innerhalb des Überwachungssystems so gestaltet sein, daß in eine Überwachungsaktivität die Daten von verschiedenen Meßsystemen einfließen können und einmal erfaßte Daten mehreren Benutzern entsprechend ihren Anforderungen zugänglich sind.

Überwachungsaktivitäten sind alle prinzipiell gleich aufgebaut. Fünf Teilfunktionen lassen sich unterscheiden:

- das Sammeln der Daten,
- das Herausfiltern relevanter Informationen,
- die Zusammenfassung der Daten von mehreren Sensoren,
- die Verarbeitung der Daten zu Überwachungsergebnissen und
- das Visualisieren bzw. Speichern der Ergebnisse.

Das Sammeln der Daten über das Netz erfolgt durch Abfragen der verteilten Sensoren sowie durch die Entgegennahme der eintreffenden Ereignismeldungen. Dabei müssen die systemspezifischen Kommunikationsprotokolle und Steuerungsmethoden berücksichtigt werden, da nicht grundsätzlich davon ausgegangen werden kann, daß die Sensoren eine Management Information Base unterstützen. Beim Zusammenfassen der Daten mehrerer Sensoren scheitern die speziellen Überwachungssysteme, die mit einigen Meßsystemen angeboten werden, da mit ihnen ein integrierter Einsatz unterschiedlicher Systeme nicht möglich ist. Es ist vielmehr ein zentrales Überwachungssystem notwendig, welches die verschiedenen Teilinformationen zu einem Datensatz zusammenfaßt. Da Sensoren unterschiedliche Meßkonzepte besitzen können, wodurch die erfaßten Werte in ihrem Format und ihrer Bedeutung variieren, ist die Korrelation dieser Daten jedoch häufig schwierig.

Ob Teilfunktionen der Netzüberwachung in den verteilten Sensoren oder im Überwachungssystem durchgeführt werden, hängt von den Komponenten ab. Das Überwachungssystem hat die Funktionalität der Sensoren entsprechend zu ergänzen und Unterschiede auszugleichen. Neben der generellen Dauerüberwachung des gesamten Netzes werden bestimmte Komponenten oder Netzbereiche vorübergehend einer intensiveren Beobachtung unterzogen, um z.B. sporadisch auftretende Fehler zu lokalisieren. Dies erfordert von einem Überwachungssystem

die Unterstützung nebenläufiger Vorgänge. Diese Vorgänge müssen weitgehend voneinander isoliert werden, um zu verhindern, daß spontane Aktivitäten wichtige, längerfristige Überwachungen beeinträchtigen. Darüber hinaus muß das Gesamtsystem eine hohe Stabilität und Robustheit aufweisen, damit Langzeitüberwachungen überhaupt erst möglich werden.

Die hier dargestellten Anforderungen stellen die Sicht des Netzbetreibers bzw. -verwalters dar. Aus der Sicht eines Benutzers kann es von Interesse sein, daß nicht alle Daten erfaßt werden, d.h. ein bestimmter Schutz gegenüber der Meßwerterfassung besteht. Da es sich hier jedoch in der Regel um private Netze handelt, spielt dieser Aspekt eine untergeordnete Rolle.

3.2 Übersicht über das neue Überwachungskonzept

Der Hauptgrund, weshalb die Überwachungskonzepte der Netzmanagementplattformen nicht in der Lage sind, heterogene Meßsysteme zu integrieren, liegt in der Festlegung der Management Information Base als Schnittstelle zwischen Meßsystem und der Plattform. Die Arbeitsweise des Netzmanagements ist datenorientiert. Somit kann eine Integration nur dann stattfinden, wenn die Managementinformationen vergleichbar sind und sich in einem Datenmodell darstellen lassen. Dies ist bei unterschiedlichen Meßsystemtypen im allgemeinen jedoch nicht der Fall, auch wenn sie für dieselben Überwachungsaufgaben eingesetzt werden.

Als Beispiel sollen drei Meßsysteme zur Erfassung der Netzlast betrachtet werden. Die Überwachungsaufgabe bestehe darin, die aktuelle Netzlast periodisch, stichprobenhaft während eines bestimmten Meßintervalls zu erfassen und aufzuzeichnen, wobei die Periodendauer wesentlich größer als das Meßintervall angenommen wird. Das erste Meßsystem sei nun so konzipiert, daß nach jeder Periodendauer eine Ereignismeldung an die Managerstation verschickt wird (Pulsar-Modus, vgl. Kapitel 2.3.5). Das zweite Meßsystem soll den jeweils aktuellen Wert lokal speichern und erfordern, daß dieser Wert periodisch abgefragt wird (vgl. RMON-Gruppe *Statistics*). Das dritte Meßsystem hingegen soll nur auf Anforderung eine Messung ausführen und das Ergebnis am Ende der Messung an den Auftraggeber liefern [127]. Das bedeutet, daß der Manager periodisch einen Meßauftrag mit entsprechender Meßdauer im Meßsystem starten muß. Solche Meßaufträge wären im Netzmanagement beispielsweise über Test-Objekte der Test Management Function realisierbar.

Dieses kleine Beispiel zeigt, daß sich Unterschiede in den Eigenschaften und den Bedienkonzepten der Meßkomponenten zwangsläufig in der Struktur der Managed Objects niederschlagen. In dem hier vorgestellten Konzept werden deshalb nicht die Daten, sondern die Überwachungsvorgänge in den Vordergrund gestellt. Bei näherer Betrachtung stellt man fest, daß diese Überwachungsvorgänge nichts anderes als Messungen darstellen, die mit denen in elektronischen Schaltungen verglichen werden können. Dort existieren Meßinstrumente, die für den Menschen nicht sichtbare Größen erfassen und an verschiedenen Stellen in der Schaltung einsetzbar sind. Jedes Meßinstrument besitzt eine Meßfunktion, wie etwa die Bestimmung der

Spannung zwischen zwei Meßpunkten. Es kann vorkommen, daß eine Meßanzeige, wie z.B. das Bild eines Oszilloskops, gleich das Ergebnis für mehrere Meßvorgänge beinhaltet (Spannung, Periodendauer und Kurvenform) oder daß für eine Messung mehrere Meßinstrumente in geeigneter Kombination eingesetzt und die Ergebnisse korreliert werden. In der Netzüberwachung werden die Meßinstrumente durch die verteilten Sensoren repräsentiert, die vielfältig in verschiedenen Überwachungsaktivitäten eingesetzt werden können. Die Messungen entsprechen den Überwachungsaktivitäten und die Meßergebnisse können mit den verarbeiteten Informationen, die der Benutzer erhält (siehe Bild 3.1), verglichen werden.

Die Modellierung der Überwachungsaktivitäten als Messungen und deren direkte Unterstützung durch das Überwachungssystem stellt den zentralen Punkt des neuen Überwachungskonzeptes dar. Messungen besitzen eine gewisse Funktionalität in Bezug auf die Erfassung und die Aufbereitung der Daten, welche über den Typ der Messung festgelegt ist. Veränderbare Eigenschaften werden über Parameter definiert und erlauben, die Messung an die Überwachungsabsicht anzupassen. Wichtige allgemeine Parameter sind der zeitliche Ablauf (siehe Kapitel 3.3), die Sensoren, welche in die Messung einbezogen sind, und Angaben zur Ergebnisdarstellung, soweit diese nicht durch den Meßtyp festgelegt sind. Der Benutzer definiert eine Messung, indem er einen Meßauftrag an das Überwachungssystem übergibt, welcher den Typ der Messung und sämtliche Parameter enthält. Das Überwachungssystem startet die Messung zum angegebenen Zeitpunkt und ist für deren Durchführung verantwortlich. Durch die Modellierung der Überwachungsvorgänge als Messungen, die von dem Überwachungssystem als Objekte behandelt werden, ergeben sich folgende Vorteile:

- Der Anwender besitzt jederzeit eine Übersicht über alle laufenden Aktivitäten und kann diese kontrollieren.
- Der Zugriffsmechanismus auf die zu erfassenden Daten und die Besonderheiten der Sensoren bleiben dem Benutzer verborgen.
- Das Überwachungssystem kann durch das Hinzufügen neuer Messungen einfach erweitert werden.
- Die Trennung von Datenerfassung, Meßfunktion und Behandlung der Ergebnisse schafft vielfältige Kombinationsmöglichkeiten.

Den allgemeinen Zusammenhang zwischen Sensor, Messung und Meßergebnis zeigt Bild 3.2. Die von den Sensoren erfaßten Daten können zum einen die Grundlage für mehrere Messungen bilden, zum anderen können mehrere Sensoren für eine Messung notwendig sein. Jedem Meßauftrag werden deshalb die Sensoren, auf denen die Messung basiert, über Meßparameter mitgeteilt. Von verschiedenen Benutzern benötigte Meßergebnisse werden durch die Messung mehrfach dargestellt. Ein Meßauftrag beschreibt eine Überwachungsfunktion aus der Sicht des Anwenders. Diese Beschreibung ist mitunter relativ abstrakt und unabhängig von den eingesetzten Sensoren.

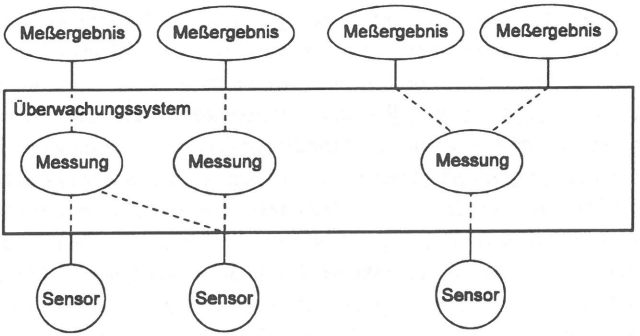


Bild 3.2: Zusammenhang zwischen Sensor, Messung und Meßergebnis

Basismeßaufträge (BMA) werden von einem Sensor direkt ausgeführt. Sie stellen die Bausteine dar, aus denen die Meßaufträge zusammengesetzt sind, und werden zwischen dem Überwachungssystem und dem Sensor übertragen. Für Sensoren, die ein standardisiertes Managementprotokoll unterstützen, entsprechen die Basismeßaufträge den GET- und SET-Requests. Die von den Sensoren erfaßten Daten stellen die Basismeßergebnisse dar. Bild 3.3 zeigt die Gliederung eines Meßauftrags und die Zusammenführung der Basismeßergebnisse zu einem Meßergebnis. In der Zusammenführung sind die Funktionen zur Verarbeitung der von den Sensoren erfaßten Daten lokalisiert. Für periodische Messungen setzt sich das Meßergebnis aus zeitlich aufeinanderfolgenden Teil- oder Zwischenergebnissen zusammen.

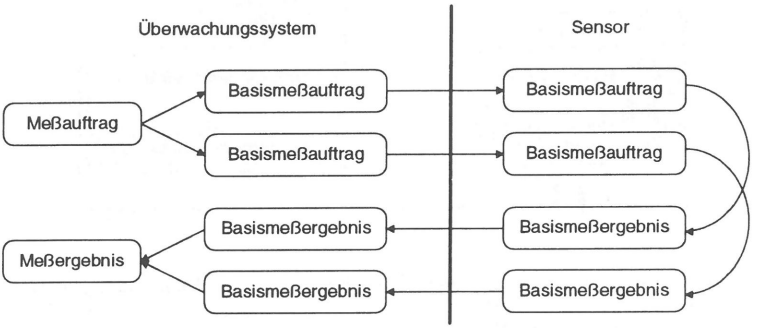


Bild 3.3: Zusammensetzung von Meßauftrag und Meßergebnis

Analog der Strukturierung von Messungen in Basismeßaufträge ist es denkbar, daß mehrere Messungen zu einer komplexeren Messung zusammengesetzt werden. Damit wird es möglich, Messungen untereinander zu korrelieren und einzelne Meßergebnisse wiederzuverwenden. Hierzu ein Beispiel: Es existiere ein Meßauftrag, der einen Echotest zu einer angebbaren Sta-

tion durchführt, welche auf demselben Ethernet-Segment wie die Meßstation liegt. Ein zweiter Meßauftrag diene der Durchführung einer Reflektionsmessung während des Betriebs. Der Echotest liefere den Wert „Wahr" für eine erreichbare Station oder „Falsch" für eine nicht erreichbare Station. Das Ergebnis einer Reflektionsmessung besteht aus einer Kurve mit der örtlichen Verteilung der Störstellen auf dem Medium. Basierend auf diesen Meßaufträgen kann eine komplexere Messung aufgebaut werden, welche periodisch Echotests zu mehreren Stationen auf einem Segment durchführt und die Ergebnisse in einer Netzkarte darstellt. Fallen nun mehrere Stationen gleichzeitig aus, wird automatisch eine Reflektionsmessung ausgeführt und in einer Nachbearbeitung der Reflektionskurve die relevanten Störstellen bestimmt. Treten solche Störstellen auf, wird ein Alarm ausgelöst und das betroffene Netzsegment in der Netzkarte gekennzeichnet. Bei Bedarf kann die komplette Reflektionskurve angezeigt werden.

Die Integration der heterogenen Meßkomponenten über Messungen desselben Typs aber mit angepaßter Realisierung kann auf zwei Ebenen erfolgen:

- Zwei Sensoren führen die gleichen BMAs über unterschiedliche Kommunikationsmechanismen aus.
- Derselbe Meßauftrag wird über unterschiedliche BMAs realisiert.

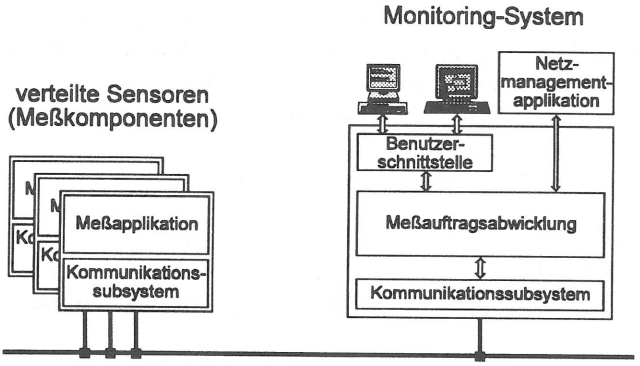


Bild 3.4: Prinzipielle Architektur und Einbettung des Monitoring-Systems

Die Umsetzung des Überwachungskonzeptes erfolgt über ein *Monitoring-System*[1] (MTS), dessen prinzipielle Architektur und dessen Einbettung in das übrige Umfeld Bild 3.4 darstellt. Es

[1] Der Begriff „Monitoring-System" wird in dieser Arbeit einheitlich für die Bezeichnung eines speziellen Überwachungssystems verwendet, welches das auf der Durchführung von Messungen beruhende Überwachungskonzept realisiert. Es ist Kapitel 4 genauer beschrieben.

besitzt für die verschiedenen Benutzerklassen angepaßte Schnittstellen. Die Benutzerschnittstelle stellt Zugänge für menschliche Anwender in Form von graphischen und zeichenorientierten Oberflächen zur Verfügung. Die Schnittstelle zum eigentlichen Kernstück, der Meßauftragsabwicklung, ist so ausgeführt, daß auch andere Netzmanagementapplikationen mit minimalem Anpassungsaufwand auf das Monitoring-System zugreifen können. Die Interaktionen beschränken sich auf die Erzeugung von Meßaufträgen, die Übermittlung bzw. Darstellung der Meßergebnisse und die Kontrolle der laufenden und zur Ausführung anstehenden Messungen. Die Meßauftragsabwicklung ist für die zeitliche Koordination und die Durchführung aller Messungen zuständig. Sie beinhaltet Meßprogramme, welche die Meßaufträge in Zusammenarbeit mit den Meßapplikationen in den verteilten Sensoren durchführen. Die Meßprogramme und Meßapplikationen sind speziell aufeinander abgestimmt. Das Kommunikationssubsystem stellt auf jeder Seite die notwendigen Protokolle und Dienste für die Übertragung der Basismeßaufträge und deren Ergebnisse zur Verfügung.

3.3 Die zeitlichen Aspekte der Messungen

Über den zeitlichen Ablauf einer Messung wird ein großer Teil der Überwachungsabsichten ausgedrückt. Er unterscheidet Hintergrundaufträge von spontanen Einzelmessungen und definiert die zeitliche Auflösung der Meßergebnisse. Eine Messung ist ein zeitlich begrenzter Vorgang, an dessen Ende ein Ergebnis zur Verfügung steht. Ein Meßauftrag kann eine einmalige oder eine periodisch wiederholte Messung spezifizieren.

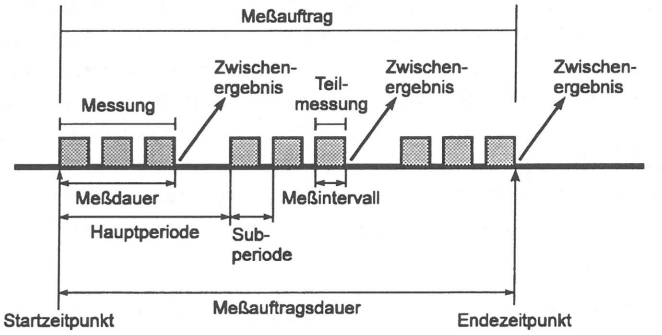


Bild 3.5: Zeitlicher Ablauf eines periodischen Meßauftrages

Bild 3.5 zeigt den allgemeinen zeitlichen Ablauf eines periodischen Meßauftrages mit den zugehörigen Parametern. Um Messungen vorzuplanen oder zueinander in eine zeitliche Beziehung zu setzen, ist es wichtig, daß für einen Meßauftrag ein Startzeitpunkt angegeben werden kann. Zeitlich begrenzte Meßaufträge enden nach der angegebenen Meßauftragsdauer oder zu

einem definierten Endezeitpunkt. Unbegrenzte Messungen müssen durch den Benutzer abgebrochen werden. Periodische Meßaufträge eignen sich für Beobachtungen über einen längeren Zeitraum, bei denen Zwischenergebnisse erforderlich sind. Periodische Abfragen, bei denen das Monitoring-System oder der Sensor selbst die erfaßten Daten zu einem Ergebnis zusammenfaßt, werden durch eine Messung, die in Teilmessungen strukturiert ist, modelliert. Dies bietet die Möglichkeit der statistischen Zusammenfassung von Daten im Rahmen einer Messung. Ob ein Basismeßauftrag einer Messung oder einer Teilmessung entspricht, ist von der Funktionalität des Sensors abhängig. Der Anwender hat jedoch grundsätzlich keinen Zugriff auf die Ergebnisse einer Teilmessung.

Die Meßparameter sind teilweise voneinander abhängig und redundant. So kann z.B. der Endezeitpunkt aus dem Startzeitpunkt und der Meßauftragsdauer berechnet werden. Der Anwender soll bei der Meßauftragsdefinition jedoch die für ihn intuitiven Parameter auswählen können und die Umrechnung dem Überwachungssystem überlassen.

3.4 Definition von Meßaufträgen

Die Meßaufträge, die der Anwender mit dem Monitoring-System durchführt, sind so vielfältig und verschieden wie die Überwachungsaktivitäten und die Netzkonfigurationen selbst. Auch wenn die Anzahl der Meßfunktionen, die ein Überwachungssystem zur Verfügung stellt, beschränkt ist, wird es dem Netzadministrator nicht erspart bleiben, die Aufträge mit den entsprechenden Parametern selbst zu definieren. Hier bietet sich der Einsatz einer speziellen Meßauftragsbeschreibungssprache an, die es dem Anwender erlaubt, Meßaufträge direkt zu formulieren und als Zeichenkette dem Monitoringsystem zu übergeben. Eine solche Meßauftragsdefinition kann natürlich auch über eine dialoggesteuerte Benutzungsoberfläche erzeugt werden. Eine textbasierte Beschreibung ist nachträglich wesentlich leichter erweiterbar als andere Schnittstellen. Für die Interpretation durch das Monitoring-System wird ein entsprechender Interpreter benötigt, der fehlerhafte Auftragsdefinitionen erkennt und korrekte Beschreibungen in eine interne Struktur konvertiert. Es existieren Werkzeuge, die, basierend auf der Sprachdefinition, die Generierung eines Programmgerüsts für einen Interpreter unterstützen [99].

Eine Sprache für die Meßauftragsbeschreibung muß folgenden Anforderungen genügen:

- leichte Erlernbarkeit durch eine für den menschlichen Benutzer gut verständliche, einfache und möglichst natürlichsprachliche Syntax,
- Abdeckung aller denkbaren Meßmöglichkeiten des Systems,
- Erweiterbarkeit und
- einfache Umsetzbarkeit in eine systeminterne Auftragsstruktur.

Für die formale Spezifikation bieten sich die Backus-Naur-Form (BNF) und Syntaxgraphen an [5], [102]. Beide definieren Regeln, mit denen syntaktisch richtige Symbolfolgen erzeugt werden können. Entscheidend für den Anwender ist, daß aus diesen Symbolfolgen die Eigenschaften des Meßauftrages intuitiv erfaßt werden können und umgekehrt nach kurzer Lernzeit der Benutzer selbst in der Lage ist, eigene Aufträge korrekt zu formulieren. Damit diese Sprache leicht interpretiert werden kann, muß sie die Eigenschaft der Kontextfreiheit besitzen und dem Prinzip des „One Symbol Lookahead, No Backtracking" gehorchen. Vereinfacht bedeutet dies, daß die Analyse einer in sich abgeschlossenen Symbolfolge unabhängig von den vorhergehenden oder nachfolgenden Symbolen möglich sein muß und daß der zu wählende Weg bei Verzweigungen im Syntaxbaum durch das erste Symbol festgelegt ist. Darüber hinaus muß sich das erste Symbol einer optionalen Folge von dem ersten nachfolgenden Symbol unterscheiden.

Die Erlernbarkeit der Sprache wird durch die einheitliche Definition gemeinsamer, allgemeiner Parameter aller Meßaufträge erleichtert. Diese sind:

- der zeitliche Ablauf der Messung,
- die involvierten Meß- und Netzkomponenten und
- Angaben zur Ergebnisdarstellung.

Viele Meßaufträge sind allein durch die Angabe des Typs und der allgemeinen Parameter vollständig definiert, andere Meßaufträge enthalten typspezifische Parameter in einer individuellen Syntax. Unzulässige Parameterkombinationen werden nicht durch die Syntax verhindert, sondern über semantische Einschränkungen, die typspezifisch für die Meßaufträge festgelegt sind. Die semantischen Einschränkungen erfordern keinen zusätzlichen Lernaufwand, da sie lediglich sinnlose bzw. in sich widersprüchliche Meßaufträge verhindern. Die meisten Parameter sind positionsunabhängig und optional. Sie besitzen Defaultwerte, die von der Meßauftragsklasse abhängen. Ein Meßauftrag besteht aus dem Schlüsselwort für den Auftragstyp und einer Parameterliste. Ein Parameter besteht entweder aus einem Schlüsselwort und möglicherweise einem Wert oder er enthält als komplexer Parameter weitere Parameter. Ein solcher komplexer Parameter beschreibt den zeitlichen Ablauf einer Messung entsprechend Kapitel 3.3. Er berücksichtigt, daß

- jede Messung periodisch durchgeführt werden kann,
- Messungen aus zeitlich versetzten Teilmessungen zusammengesetzt sein können, die zu einem Ergebnis zusammengefaßt werden,
- viele Messungen ein benutzerdefiniertes Meßintervall benötigen.

Die Meßauftragsbeschreibungssprache erlaubt auf unterschiedliche Art und Weise den zeitlichen Ablauf eines Meßauftrages festzulegen. Dies ergibt sich zum einen aus der Redundanz in den Parametern „Hauptperiode", „Anzahl der Messungen" und „Meßdauer", zum anderen durch die Freiheit, Zeitpunkte absolut oder relativ zum gegenwärtigen Zeitpunkt anzugeben.

Dabei sind die Spezifikation von Zeitpunkten und Zeitdauern an die Programmiersprache für Echtzeitsysteme PEARL (Process and Experiment Automation Real-time Language) [23] angelehnt. Folgende Beispiele für den komplexen Parameter TIME[] zeigen die Flexibilität bei den Angaben und die leichte Verständlichkeit:

`TIME[AT 7:45, ALL 3 MIN, UNTIL 18:30]`
Start um 7.45 Uhr, Hauptperiode 3 Minuten, Ende gegen 18.30 Uhr - ergibt eine Meßauftragsdauer von 10 Stunden 45 Minuten und 215 Messungen.

`TIME[AT 8.5.95 8:00, DURING 10 HRS, 60 TIMES]`
Start am 8. Mai 1995 um 8.00 Uhr, Meßauftragsdauer 10 Stunden, 60 Messungen - ergibt ein Ende der Messung um 18.00 Uhr und eine Hauptperiode von 10 Minuten.

`TIME[AFTER 30 MIN, ALL 20 SEC, 15 TIMES, INTERVAL 2 SEC]`
Start in 30 Minuten, Hauptperiode 20 Sekunden, 15 Messungen, Meßdauer 2 Sekunden - ergibt eine Meßauftragsdauer von 5 Minuten.

`TIME[SUBTIME[ALL 1 SEC, 60 TIMES, RESULT MEAN]]`
Einmalige Messung, die sofort startet, Subperiode 1 Sekunde, 60 Teilmessungen, wobei das Zwischenergebnis als Mittelwert aus den Teilmessungen gebildet wird - die Meßdauer ist gleich der Meßauftragsdauer und beträgt 1 Minute.

Für die Festlegung der an einer Messung beteiligten Stationen müssen die Stationen selbst bezeichnet und ihre Beziehungen zueinander ausgedrückt werden. Es wird zwischen primären und sekundären Stationen unterschieden. Primäre Stationen sind die eigentlichen Meßkomponenten, die einen Basismeßauftrag durchführen. Messungen mit Testcharakter, wie z.B. ein Echotest, beziehen sich auf bestimmte Stationen im Netz, die als sekundäre Stationen bezeichnet werden. Die Identifikation der Stationen erfolgt über ihren Namen, ihre Netzadresse oder über ihre Zugehörigkeit zu einer Stationsgruppe. Die Konfigurationsdatenbank speichert die Zusammenhänge zwischen diesen Angaben. Stationsgruppen erlauben dem Benutzer in einfacher Weise netzweite Messungen durchzuführen, ohne sich um die Meßkomponenten in den verschiedenen Netzbereichen Gedanken machen zu müssen. Ein weiterer Parameter ist zur Unterscheidung der an einer Messung beteiligten Meßapplikation vorgesehen, falls diese nicht aus dem Meßauftragstyp abgeleitet werden kann. Wird ein TCP/IP-Protokoll verwendet, kann hier die Port-Nummer der Meßapplikation eingetragen werden. Schließlich kann noch eine Liste mit Protokollen für die Kommunikation mit der betreffenden Station spezifiziert werden. Für primäre Stationen bezeichnet diese Liste die für den Meßauftrag möglichen Transportprotokolle, bei einer sekundären Station gibt die Liste an, mit welchen Protokollen der Test zu dieser Station durchgeführt werden soll. Über das Symbol „->“ wird eine primäre Stationsliste mit einer sekundären Stationsliste verknüpft. Dabei ist jedem Element der primären Liste die

komplette sekundäre Liste zugeordnet. Bild 3.6 zeigt den Syntaxgraphen [82] für den komplexen Stationsparameter STAT[].

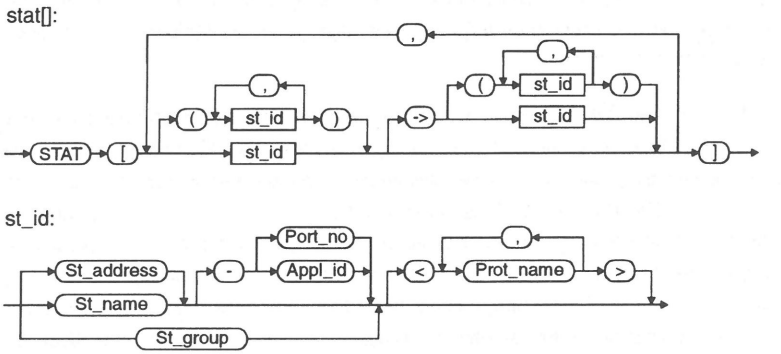


Bild 3.6: Syntaxdiagramm des komplexen Stationsparameters

Die Angaben zur Ergebnisdarstellung definieren hauptsächlich, wie Zwischenergebnisse bei periodischen Messungen gehandhabt werden. Es existieren vier Modi:

- Save_All: Alle Ergebnisse werden gespeichert, d.h. die Historie ist abrufbar. Die Ausgabe wird durch den Benutzer angestoßen.
- Overwrite: Das aktuelle Ergebnis wird gespeichert und steht jederzeit zur Verfügung. Alte Ergebnisse werden überschrieben.
- Queue: Alle Ergebnisse werden selbständig ausgegeben, wobei sie in einer Warteschlange zwischengespeichert werden, falls Verzögerungen bei der Ausgabe auftreten.
- Queue_And_Save: Wie Queue, nur daß die Ergebnisse zusätzlich auch gespeichert werden und durch den Benutzer abfragbar sind.

Ergänzt werden die allgemeinen Parameter um meßauftragsspezifische Parameter, mit denen zusätzliche Angaben zur Messung gemacht werden können. Eine komplette Beschreibung der Sprachsyntax, wie sie in dem in Kapitel 4 vorgestellten Prototyp realisiert ist, befindet sich im Anhang. Einige der dort aufgeführten Parameter wurden hier nicht beschrieben, da sie Angaben betreffen, die spezifisch für die Architektur des Prototyps sind.

3.5 Netz- und Meßsystemkonfiguration

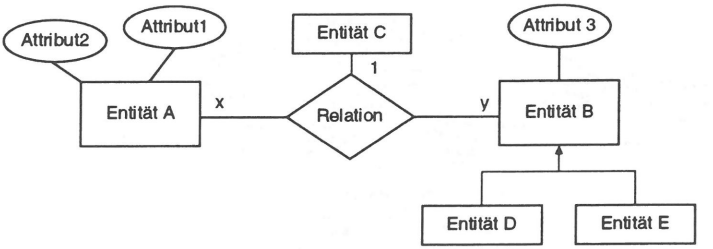
Für die Bezeichnung der Meßobjekte und der Meßkomponenten eines Meßauftrages benötigt der Anwender des Monitoring-Systems primär deren Namen, Netzadressen und Standorte. Bei Sensoren sind zusätzlich die Ausstattung bzw. die mit dieser Komponente durchführbaren Messungen und deren aktueller Status interessant. Darüber hinaus sind Informationen über die

Netztopologie und Angaben zu den übrigen Netzkomponenten hilfreich. Bei der Definition einer Messung sollte für die Festlegung einer Meßstation das zu überwachende Objekt (z.B. ein Ethernet-Segment) ausreichen und das Monitoring-System sollte anhand der ihm zur Verfügung stehenden Konfigurationsinformationen eine passende Meßkomponente auswählen können.

Beim Einsatz einer Netzmanagementplattform werden diese Informationen dort zumindest teilweise im Rahmen des Configuration Managements bereits erfaßt und in einer sogenannten Managerdatenbank gespeichert. Diese Datenbank bildet die Grundlage zur Darstellung der Netztopologie. Die Ermittlung dieser Informationen erfolgt weitgehend automatisch, indem, beginnend mit einer Station, Adreßinformationen über deren Kommunikationspartner über die Managementschnittstelle ausgelesen und als neue Stationen der Netzkonfiguration hinzugefügt werden. Daraufhin wird die Konfiguration dieser Stationen abgefragt. Dieses Vorgehen wird rekursiv fortgesetzt und führt zu einer ringförmigen Erweiterung des Wissens über das Netz [37].

Das Monitoring-System benötigt für seine Arbeit nur einen Teil der Informationen der Managerdatenbank. Diese müssen jedoch so strukturiert sein, daß Fragen an die Netztopologie, wie z.B. welche Segmente zu einem Teilnetz gehören oder welche Koppelelemente auf dem Kommunikationspfad zwischen zwei Stationen liegen, beantwortet werden können. Auch muß es möglich sein, die für die Erfassung bestimmter Netzgrößen zuständigen Sensoren und deren Belegungszustand zu bestimmen, wofür unter Umständen zusätzliche Angaben benötigt werden. Damit die Konfigurationsinformationen für das Monitoring-System von Meßprogrammen aus benutzbar sind, ist eine eigene, von der Managementplattform unabhängige, Datensicht notwendig.

Die Modellierung einer Datensicht resultiert in einem konzeptionellen Entwurf dieser Daten, welcher, unabhängig von der Realisierung, die Daten selbst und ihre Beziehungen untereinander definiert. Für die formale Beschreibung eines solchen Entwurfs ist das Entity-Relationship-Modell (ER-Modell) weit verbreitet [6], [17]. Es besteht aus Entitätsmengen und Relationsmengen. Entitätsmengen modellieren Datenobjekte eines Typs. Eine Relationsmenge beschreibt eine Beziehung zwischen zwei Entitätsmengen, die optional durch eine dritte Entitätsmenge charakterisiert sein kann. Die Eigenschaften von Entitäten und Relationen bzw. deren Datenelemente werden durch Attribute modelliert. Zur Darstellung von ER-Modellen werden in der Literatur leicht unterschiedliche Notationen verwendet. Bild 3.7 zeigt die für diese Arbeit verwendete Darstellungsform. Sie basiert auf der Notation nach Chen [17], die um den Beziehungstyp *Generalisierung* bzw. *Verfeinerung* (Pfeil) erweitert ist.



Die Entität C ist der Relation zwischen Entität A und Entität B zugeordnet.

Die Entitäten D und E sind Verfeinerungen der Entität B („ist eine"-Beziehung) bzw. Enitität B ist eine Generalisierung von D und E.

Kardinalitäten: Eine Instanz der Entität A steht in Relation zu y Instanzen der Entität B.
Eine Instanz der Entität B steht in Relation zu x Instanzen der Entität A.

Bild 3.7 Darstellung von Entitäten und Relationen nach Chen

Der konzeptionelle Entwurf der Konfigurationsdatenbank für das Monitoring-System ist in Bild 3.8 dargestellt und orientiert sich primär an den für das Monitoring-System benötigten Funktionen bei der Auftragserzeugung und der Meßdurchführung. Diese sind:

- die Zuordnung von Namen der Meßkomponenten zu Netzadressen und umgekehrt,
- die Abfrage der Konfiguration und des Status der Meßkomponente und
- die Belegung der Meßkomponente für die Dauer eines Basismeßauftrags.

Für die automatisierte Erstellung von Meßaufträgen bzw. die Darstellung der Netzkonfiguration ist es wichtig, die hierarchische Strukturierung eines Netzes in Teilnetze, Segmente, Netzkoppelelemente und Netzkomponenten (Netzeinheiten) zu berücksichtigen und deren Topologie zu speichern. Die Meßkomponenten sind als spezielle Netzeinheiten in die Hierarchie eingegliedert. Meßkomponenten bestehen aus Meßapplikationen, die wiederum in der Lage sind, Basismeßaufträge auszuführen. Die Beziehung zwischen den benutzerdefinierten Meßaufträgen und den Basismeßaufträgen, aus denen sich ein Meßauftrag zusammensetzt, ist in dem Konfigurationsdatenmodell enthalten, damit die für die Durchführung in Frage kommenden Meßkomponenten ermittelt werden können bzw. intern geprüft werden kann, ob ein Meßauftrag mit den angegebenen Meßstationen überhaupt durchführbar ist. Damit die Erzeugung der Meßaufträge nicht wesentlich verzögert wird, muß der Zugriff auf diese Daten sehr schnell erfolgen.

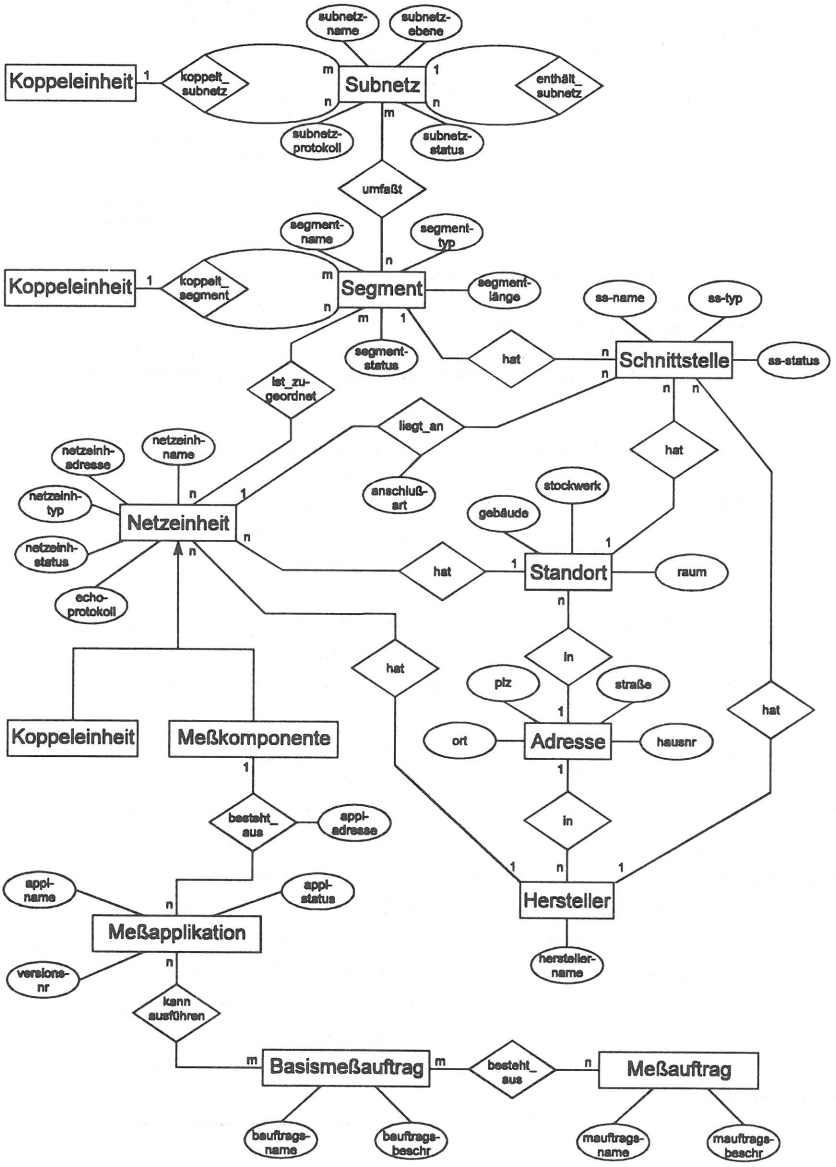


Bild 3.8 Konzeptionelles Schema der Konfigurationsdaten des Monitoring-Systems

3.6 Einordnung des Monitoring-Systems in die Managementarchitektur

Ein vollständig integriertes Netzmanagement, in welchem alle verwendeten Managementapplikationen auf einer Plattform arbeiten und mit dem alle Managementaufgaben in heterogenen Netzen abgedeckt werden, ist mittel- und langfristig nicht verfügbar. Netzmanagementlösungen stellen einen Kompromiß dar zwischen dem technisch Wünschenswerten, den auf dem Markt verfügbaren Produkten und dem finanziellen Aufwand. Die Grundfunktionalität des Monitoring-Systems, Messungen durchzuführen und zu verwalten, kann abhängig von der Gestalt der Managementlösung an verschiedenen Stellen angesiedelt sein. Anzupassen ist dabei lediglich die Schnittstelle zum Anwender.

Häufig besteht der Bedarf nach einer direkten Nutzung der Netzüberwachung durch menschliche Anwender, zum Teil auch in Bereichen, in denen kein anderes Managementsystem vorhanden ist. In diesem Fall arbeitet das Monitoring-System eigenständig und muß eine vollständig ausgebaute Benutzerschnittstelle besitzen sowie minimale Konfigurationsinformationen zur Verfügung stellen. Es ersetzt, wie in Bild 3.9 dargestellt, die verschiedenen, unabhängig voneinander eingesetzten Überwachungswerkzeuge, indem es deren Funktionen als Messungen in einem System unter einer einheitlichen Benutzeroberfläche integriert.

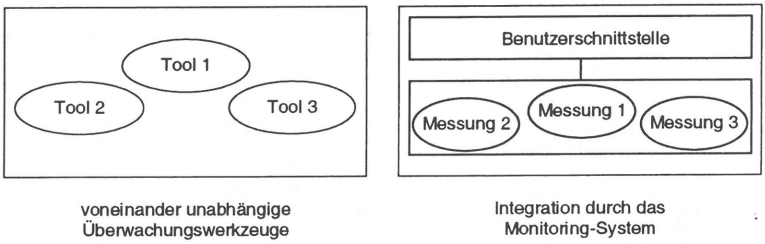


Bild 3.9: Einsatz des Monitoring-Systems als eigenständiges System

Einige Managementapplikationen legen ihren Schwerpunkt auf die Informationsverarbeitung. Dabei werden u.a. Methoden der künstlichen Intelligenz (KI), der Fuzzy-Logik oder neuronale Netze verwendet. Bei diesen Applikationen tritt die Informationserfassung in den Hintergrund und muß oft vom Netzadministrator zur Verfügung gestellt werden. Als Beispiel soll hier ein System für die wissensbasierte Diagnose in lokalen Netzen [128] betrachtet werden. Tritt in einem Netz eine Fehlfunktion auf, kann über die Bewertung von Symptomen, welche das Diagnosesystem in einem Dialog mit dem Benutzer einholt, die Fehlerursache ermittelt werden. Die abzufragenden Symptome bestimmt das System mit Hilfe von Regeln der KI und einer Wissensbasis mit Informationen über das Netz. Anstelle des Benutzerdialogs ist es teilweise sinnvoll, automatische Symptombewertungsmethoden zu verwenden, welche die benötigten Informationen aus dem Netz beziehen. Hier bietet es sich an, die Informationsbeschaffung in

das Monitoring-System auszulagern, indem das Diagnosesystem die Anwenderrolle einer Netzmanagementapplikation einnimmt.

Das Diagnosesystem ist hierzu mit einer sehr einfachen Monitoring-Schnittstelle ausgestattet. Diese besteht darin, daß für die Symptombewertung ein beliebiges in der Wissensbasis definiertes Programm gestartet wird, welches die Bewertung als Integer-Zahl zurückliefert. Startet dieses Programm einen Meßauftrag, dessen Ergebnis auf diese Zahl reduziert wird, kann die automatische Symptombewertung über das Monitoring-System gleichzeitig mit anderen Messungen erfolgen.

Für eine Integration des Monitoring-Systems in ein bestehendes Netzmanagementsystem bieten sich zwei verschiedene Formen an, die in Bild 3.10 und Bild 3.11 verdeutlicht sind. Die erste Form beruht darauf, die Messungen als Managed Objects zu modellieren und das Monitoring-System als einen besonderen Agent zu betrachten.

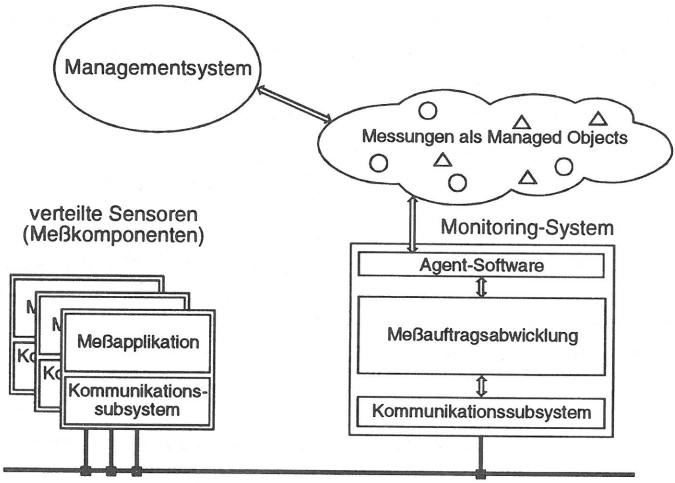


Bild 3.10: Modellierung der Messungen als Managed Objects

Hierfür bieten sich die Mechanismen der Test Management Function an (vgl. Kapitel 2.5.3). Das asynchrone Testmodell erlaubt es, Messungen in Test Objects zu kapseln, wobei die Meßparameter und Ergebnisse in Attributen abgelegt werden. Den verschiedenen Meßfunktionen können entsprechende Test Object-Klassen zugeordnet werden, die jedoch zuvor gemäß den GDMO-Richtlinien definiert werden müssen. Das Monitoring-System selbst entspricht dem TARR-Objekt, welches auch in der Lage ist, zusammengehörige Testobjekte (Messungen) zu verwalten. Die zu verwendenden Meßkomponenten können über die Managed Objects under

Test festgelegt werden. Die Steuerung des zeitlichen Ablaufes erfolgt über das Scheduling Package. Somit kann das allgemeine Modell einer Messung vollständig auf das der Managed Objects abgebildet werden.

Der Nachteil, Messungen als Managed Objects zu modellieren, besteht jedoch darin, daß entsprechende Testobjektklassen, die für das Monitoring-System definiert werden, in das Managementsystem eingebunden werden müssen. In wieweit existierende Managementapplikationen mit diesen Objektklassen sinnvoll umgehen können, ist beim gegenwärtigen Stand der Technik jedoch fraglich. Dies wird durch die Dynamik der Testobjekte verstärkt, da ein Manager a priori wissen muß, an welcher Stelle in einem Containment Tree eine Instanz einer bestimmten Objektklasse erzeugt werden darf. Diese Informationen sind lediglich in der Spezifikation des Agents in nicht formaler Form enthalten. Der Agent kann dem Manager jedoch Schablonen für instanziierbare Objekte, sogenannte Template-Objekte, anbieten, so daß innerhalb des Containment Trees eines Agents gesucht werden kann, ob der Agent bestimmte Messungen unterstützt.

In der zweiten Form ist das Monitoring-System in eine Netzmanagementplattform eingebettet. Damit ist die Verwendung einer gemeinsamen Benutzerschnittstelle und eines einheitlichen Meldungskonzeptes sichergestellt. Falls benötigt, ist ein ortstransparenter und protokollunabhängiger Zugriff auf die Management Information Base vorhanden. Bild 3.11 zeigt die prinzipielle Architektur.

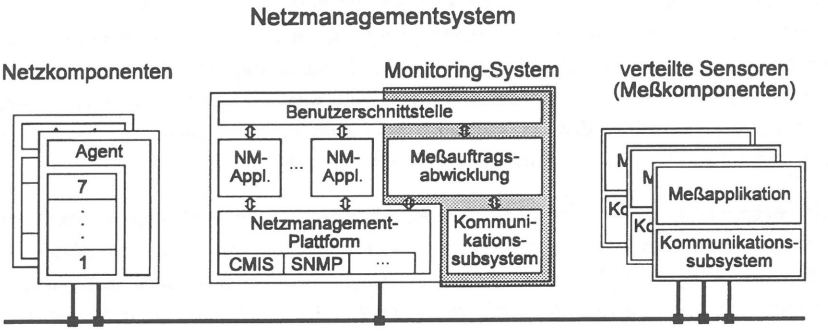


Bild 3.11: Integration des Monitoring-Systems in eine Netzmanagementplattform

Der Vorteil dieser Einbindung ist, daß das Monitoring-System auf die Infrastruktur der Netzmanagementplattform und damit auf die Konfigurationsinformationen zugreifen und somit Informationen direkt in der dortigen Netzkarte darstellen kann. Darüber hinaus wird das Abfragen von Netzkomponenten über die Netzmanagementschnittstellen stark vereinfacht. Für

die Kommunikation mit nicht standardkonformen Meßkomponenten wird weiterhin das Kommunikationssubsystem verwendet.

3.7 Benutzerschnittstelle

Die flexiblen Einsatzmöglichkeiten des Monitoring-Systems beruhen neben der anwenderspezifischen Ergebnisaufbereitung auf den flexiblen Gestaltungsmöglichkeiten der Benutzeroberfläche. Bei einer direkten Benutzung eignet sich für die übersichtliche Darstellung der voneinander unabhängigen, nebenläufigen Messungen am besten eine graphische Oberfläche. Solche Oberflächen sind fensterorientiert und erlauben, über jedes Fenster einen separaten Kommunikationskanal zum Benutzer herzustellen, um darüber Meßaufträge zu empfangen oder Ergebnisse darzustellen. Die Steuerungsinitiative liegt beim Benutzer, der beliebig zwischen den Fenstern wechseln und begonnene Dialoge unterbrechen kann. Die Messungen selbst können zur besseren Übersicht als graphische Symbole in einem Kontrollfenster dargestellt und manipuliert werden. Auch ist eine graphische Ergebnisdarstellung wesentlich schneller erfaßbar als Zahlen- oder Zeichenkolonnen. Dabei spielen verschiedene Diagrammformen und Algorithmen zur übersichtlichen Anordnung der Ergebnisse eine wichtige Rolle [41], [105]. Graphische Oberflächen erfordern jedoch entsprechende Terminals und besitzen einen relativ hohen Bedarf an Rechenleistung. Wird von einer entfernten Stelle auf das Monitoring-System zugegriffen, fällt zusätzlich die hohe auszutauschende Datenmenge ins Gewicht.

Besteht die Verbindung zum Monitoring-System lediglich aus einem seriellen, niederbitratigen Kanal, muß auf zeichenorientierte Oberflächen ausgewichen werden. Semigraphische Fensteroberflächen haben den Vorteil, daß trotz begrenzter Kommunikationsbandbreite unterschiedliche Dinge gleichzeitig auf einem Bildschirm dargestellt werden können. Manche Oberflächen jedoch sind so ausgelegt, daß die Bedienung programmgesteuert erfolgt, d.h. der Benutzer überwiegend auf Abfragen des Programmes reagiert [91], [98].

Eine weitere mögliche Benutzerschnittstelle ist die eines Kommandointerpreters (Shell), welcher die Formulierung von Meßaufträgen und die Abfrage von Ergebnissen erlaubt. Die Hauptanwendung dieser Schnittstelle liegt in der Möglichkeit des Anwenders, Shell-Skripte zu erstellen, die Routinearbeiten automatisieren oder komplexere Meßaufträge darstellen. Unter dem Betriebssystems UNIX kann dessen Standard-Shell direkt verwendet werden, die ausreichende Variablenkonzepte und Elemente für die Ablaufsteuerung besitzt [29]. Die Schnittstelle zum Monitoringsystem ist dabei über spezielle Programme zu realisieren, die von der Shell aus aufgerufen werden. Es existieren jedoch auch sehr mächtige, erweiterbare Kommandointerpreter wie Tcl (Tool Command Language) [107], [139].

Kapitel 4

Ein offenes Monitoring-System für die Überwachung von Rechnernetzen

Durch das in diesem Kapitel vorgestellte Monitoring-System erfährt das auf Messungen basierende Überwachungskonzept seine praktische Umsetzung. Damit das Monitoring-System der geforderten Erweiterbarkeit und der Nebenläufigkeit von Überwachungsaktivitäten gerecht wird, ist es als ein kooperierendes Mehrprozeßsystem ausgeführt. Ein Multi-Tasking-Betriebssystem übernimmt dabei die Prozeßverwaltung, die Ablaufkontrolle und die Kommunikation zwischen verschiedenen Prozessen. Kapitel 4.1 behandelt einige Grundlagen zu Mehrprozeßsystemen, die für das Nachvollziehen der getroffenen Designentscheidungen bedeutend sind. Die übrigen Kapitel beschreiben das Monitoring-System mehr im Detail, wobei der Schwerpunkt auf der Systemarchitektur liegt. Abschließend wird auf den zur praktischen Erprobung der Tragfähigkeit dieses Ansatzes implementierten Prototyp eingegangen.

4.1 Grundlagen zu Mehrprozeßsystemen

Die Leistung eines Rechnersystems wird durch das Ablaufen vieler Einzelschritte, sogenannter *Aktivitäten,* erbracht, die durch bestimmte Anweisungen und Eingaben gesteuert werden [34]. Die Anweisungen sind in einem Programm zusammengefaßt, welches sich zur Ausführungszeit im Hauptspeicher des Rechnersystems befindet. Der Begriff *Prozeß* im Zusammenhang mit Softwaresystemen bezeichnet die sequentielle Folge von Aktivitäten, durch die eine in sich abgeschlossene Aufgabe bearbeitet wird. Die Frage, was unter einer Aktivität oder einer in sich abgeschlossenen Aufgabe zu verstehen ist, hängt von der Abstraktionsebene ab. Im folgenden sollen unter einer Aktivität eine Prozessorinstruktion bzw. unter einer abgeschlossenen Aufgabe die Ausführung eines ganzen Programms verstanden werden.

4.1.1 Verwaltung von mehreren Prozessen auf einem Prozessor

Vernachlässigt man speicherresidente Programme, die aufgrund einer speziellen Tastenkombination ein aktuell ablaufendes Programm unterbrechen, und Interrupt-Routinen, befindet sich bei einem Einprogrammbetriebssystem (*Single-Tasking*) zu einer Zeit nur ein Programm im Speicher und wird von einem Prozessor ausgeführt. Bei entsprechender Programmierung können jedoch in einem Programm durchaus mehrere Aktivitäten und somit mehrere Prozesse quasi-parallel ausgeführt werden.

Mehrprogrammbetriebssysteme (*Multi-Tasking*) zeichnen sich dadurch aus, daß mehrere Prozesse quasi-gleichzeitig auf einem Prozessor ablaufen können, ohne daß dies vom Programmierer explizit vorgesehen ist. Die Prozesse werden vom Betriebssystem verwaltet und abwechslungsweise dem Prozessor zugeteilt. Für einen Prozeßwechsel unterbricht eine Betriebssystemkomponente, der *Scheduler*, den momentan aktiven Prozeß und läßt einen anderen Prozeß an der zuvor unterbrochenen Stelle weiterlaufen. Damit der Scheduler selbst zur Ausführung kommt, muß der laufende Prozeß die Kontrolle über den Prozessor abgeben. Man unterscheidet dabei zwischen:

- dem kooperativen Multi-Tasking (*non pre-emptive scheduling*):
  Jeder Prozeß gibt nach relativ kurzen Ausführungszeiten die Kontrolle „freiwillig" an den Scheduler ab. Diese Abgabe kann auch implizit durch den Aufruf von Systemfunktionen erfolgen. Prozesse, welche, z.B. aufgrund einer Endlosschleife, die Kontrolle überhaupt nicht mehr abgeben, führen zu einer Blockierung des Systems.

- dem Zeitscheibenverfahren (*pre-emptive scheduling*):
  Jeder Prozeß darf den Prozessor maximal eine gewisse Zeit belegen. Am Ende der Zeitscheibe erhält der Scheduler die Kontrolle über den Prozessor durch einen zeitgesteuerten Hardware-Interrupt (Timer). Es ist möglich, daß ein Prozeß die Zeitscheibe vorzeitig freigibt, wenn er einen Zustand erreicht, in dem er nicht mehr weiterarbeiten kann, z.B. beim Warten auf Eingaben oder ein momentan belegtes Betriebsmittel. Eine Blockierung des Gesamtsystems durch einen Prozeß kann nicht auftreten.

Moderne Betriebssysteme bieten zur Unterstützung unterschiedlicher Kopplungsgrade zwischen den Prozessen zwei Klassen an:

- schwergewichtige Prozesse (*heavy weight processes*) und
- leichtgewichtige Prozesse (*light weight processes, threads*).

Schwergewichtige Prozesse werden durch ein Multi-Tasking-Betriebssystem weitgehend voneinander isoliert, um so eine ungewollte, gegenseitige Beeinflussung zu vermeiden. Zugriffskollisionen auf gemeinsame Betriebsmittel werden transparent für die Prozesse vom Betriebssystem aufgelöst. Durch ein virtuelles Speicherkonzept sind den Prozessen individuelle

Bereiche für Programmcode und Daten zugewiesen. Bei einer Prozeßteilung (*fork*) wird der gesamte Datenbereich kopiert. Überschreitet ein Prozeß den ihm zugewiesenen Adreßbereich, führt dies in der Regel zu einem Programmabbruch. Der Datenaustausch zwischen verschiedenen Prozessen erfolgt durch spezielle Mechanismen des Betriebssystems.

Leichtgewichtige Prozesse (*threads*) eignen sich für Mehrprozeßsysteme, die auf demselben Programmcode basieren und bei denen der Aufwand für die Prozeßerzeugung minimal gehalten werden soll. Threads besitzen einen gemeinsamen Speicherbereich für statische Daten (z.B. globale Variablen), auf dem sie unabhängig voneinander arbeiten. Dadurch entfällt bei ihrer Erzeugung der Kopiervorgang des Datenbereichs, birgt jedoch die Gefahr von unbeabsichtigten Seiteneffekten. Eine besondere Bedeutung besitzen Threads in der Programmierung von Anwendungen mit grafischen Benutzeroberflächen, da sie die Abkopplung von größeren Rechenaktivitäten erlauben, während die Hauptaktivität schnell wieder auf Interaktionen des Benutzers reagieren kann.

4.1.2 Synchronisationsmechanismen

Bei der Programmierung von Mehrprozeßsystemen muß davon ausgegangen werden, daß ein Prozeß nach jedem Einzelschritt für eine unbestimmte Zeit unterbrochen werden kann, also auch während des Zugriffes auf gemeinsam benutzte Daten. Ohne entsprechende Synchronisationsmechanismen führt dies im günstigen Fall zu einer Nichtdeterminiertheit des Prozeßablaufes, bei dem je nach der zufälligen Reihenfolge der konkurrierenden Aktivitäten (*race condition*) das Ergebnis zwar variiert, aber gültig ist. Im ungünstigen Fall können jedoch Dateninkonsistenzen auftreten, die zu Fehlfunktionen führen.

Konkurrieren mehrere Prozesse um gemeinsame Betriebsmittel oder Daten, müssen die Zugriffe sequentialisiert werden. Auch der bei einer Prozeßkooperation notwendige Nachrichtenaustausch erfordert eine zeitliche Ordnung der verschiedenen Aktivitäten. Man unterscheidet zwei Synchronisationsarten, die einseitige Synchronisation oder Bedingungssynchronisation (*condition synchronisation*) und die mehrseitige Synchronisation, auch gegenseitiger Ausschluß (*mutual exclusion*) genannt.

Die einseitige Synchronisation gewährleistet eine bestimmte zeitliche Abfolge zweier Aktivitäten in verschiedenen Prozessen, wobei die nachfolgende Aktivität keinen Einfluß auf die vorausgehende haben darf. Sie ist anzutreffen bei unidirektionalen Datentransfers über einen gemeinsamen Speicherbereich, bei dem Prozeß A die Daten zuerst schreiben muß, bevor sie von Prozeß B ausgelesen werden können. Sollen Daten mehrfach übertragen werden, ist zu verhindern, daß Prozeß A die Daten überschreibt, bevor Prozeß B sie liest und daß Prozeß B dieselben Daten mehrfach liest. Dies wird durch eine Verkettung von einseitigen Synchronisationen erreicht. Bezeichnen „A_i“ die Schreibaktivitäten des Prozesses A und „B_i“die Leseak-

tivitäten des Prozesses B und werde die einseitige Synchronisation durch das Symbol „→" dargestellt, ergibt sich folgende Ablaufbeziehung:

$$A_i \rightarrow B_i \rightarrow A_{i+1} \qquad (4.1)$$

Die Realisierung der einseitigen Synchronisation zwischen zwei Prozessen ist relativ einfach, wenn man davon ausgehen darf, daß Schreib- und Lesezugriffe auf eine einfache, gemeinsame Variable S unteilbar sind (z.B. weil sie durch eine Assemblerinstruktion ausgeführt werden). Eine solche Variable wird auch als *Schloßvariable* bezeichnet. Eine mögliche Zugriffsprozedur (*Schloßalgorithmus*) für die einseitige Synchronisation zeigt Bild 4.1. Bei der Verkettung von einseitigen Synchronisationen ist es wichtig, daß die Schloßvariable immer im Wechsel beschrieben wird, d.h. ein Prozeß warten muß, bis der andere die Variable auf einen bestimmten Wert setzt.

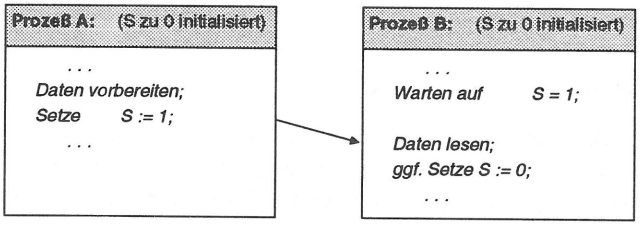


Bild 4.1: Realisierung der einseitigen Synchronisation mit Hilfe einer Schloßvariable

Sollen bestimmte Aktivitäten (kritische Abschnitte) zueinander zeitlich versetzt ablaufen, ohne daß eine Reihenfolge vorgegeben ist, wird dies als mehrseitige Synchronisation bezeichnet. Typische Beispiele sind konkurrierende Zugriffe auf ein gemeinsames Betriebsmittel oder einen gemeinsamen Datenbereich. Die Realisierung einer mehrseitigen Synchronisation über eine Schloßvariable ist zwar möglich [34], gestaltet sich jedoch wesentlich komplizierter als bei der einseitigen Synchronisation, da neben der Eigenschaft des gegenseitigen Ausschlusses auch noch die der Verklemmungsfreiheit und der Aushungerungsfreiheit einzelner Prozesse sichergestellt sein muß. Praktisch relevant sind deshalb nur Algorithmen, die auf unteilbaren Mehrfachoperationen wie *test_and_set(a, value)* basieren, bei denen ein Prozeß zwischen einem Lese- und einem Schreibzugriff nicht unterbrochen werden kann. Eine Realisierung kann aus Bild 4.2 entnommen werden.

Prozese A und B: (S zu 0 initialisiert)

```
. . .
repeat
   Warten auf S = 0;
   i = test_and_set (S, 1)     ; falls i = 1, anderer Prozeß war schneller -> warten
until i = 0;
Kritischer Abschnitt;
Setze S := 0;
. . .
```

Bild 4.2: Realisierung der mehrseitigen Synchronisation

Schloßalgorithmen besitzen prinzipiell den Nachteil, daß während des Wartens die Schloßvariable zyklisch abgefragt werden muß und somit Rechenzeit verbraucht wird (*busy waiting*). Dies wird durch vom Betriebssystem angebotene Synchronisationsmechanismen vermieden, die den Prozeß während der Wartephase blockieren und ihn erst danach wieder in den ablauffähigen Zustand versetzen. Die wichtigsten Mechanismen zur reinen Prozeßsynchronisation sind

- Signale und
- Semaphore.

Signale sind dafür konzipiert, einen Prozeß asynchron über ein unerwartetes Ereignis zu informieren, wie beispielsweise den Wunsch des Benutzers nach Programmabbruch, einen Speicherzugriffsfehler während der Programmausführung oder die Mitteilung über die Beendigung eines Kindprozesses. Es existiert jedoch auch ein Betriebssystembefehl, mit dem ein Prozeß bis zum Eintreffen eines Signals pausieren kann. Empfängt ein Prozeß ein Signal, unterbricht er seine momentane Aktivität und springt zu einer speziellen Funktion, dem *Signal Handler*, nach deren Beendigung das Programm an der zuvor unterbrochenen Stelle weitergeführt wird. Der Signal Handler entspricht somit einer Interrupt-Routine, also einer nebenläufigen Aktivität innerhalb desselben Prozesses, mit allen damit verbundenen Gefahren der Dateninkonsistenz. Signale sind die einzige Möglichkeit, einen Prozeß an einer Stelle zu unterbrechen, an der dies der Programmierer nicht vorgesehen hat. Sie sind jedoch verlustbehaftet, d.h. der Empfänger kann ein Signal ignorieren, ohne daß dies der Sender erkennen kann. Deshalb sind Signale zur Prozeßsynchronisation denkbar ungeeignet. Ausnahmen sind:

- das Warten auf die Beendigung eines Kindprozesses und
- der Abbruch eines blockierenden Systemaufrufes durch ein Timeout-Signal.

Der Begriff des Semaphors (engl. semaphore) als ein Mechanismus zur Synchronisation nebenläufiger Prozesse unter Vermeidung des aktiven Wartens wurde von Dijkstra geprägt. Ein Semaphor bezeichnet einen abstrakten Datentyp, der im wesentlichen aus einem Zähler besteht

und für den zwei Operationen definiert sind – P wie „passeeren" (passieren) und V wie „vrijgeven" (freigeben, verlassen). Die P-Operation erniedrigt den Zähler in einer nicht unterbrechbaren Operation, falls der Zähler zuvor größer Null war. Würde der Zähler durch die P-Operation negativ, wird der Prozeß blockiert und in eine dem Semaphor zugeordnete Warteschlange eingereiht. Beendet wird die Blockierung, indem ein anderer Prozeß den Zähler durch eine ebenfalls nicht unterbrechbare V-Operation erhöht. An einem Semaphor wartende Prozesse werden streng nach dem FIFO-Prinzip (*First In, First Out*) abgehandelt. Mit Semaphoren können sowohl einseitige, als auch mehrseitige Synchronisationen realisiert werden. Mehrwertige Semaphore ermöglichen die gleichzeitige Anwendung von P- und V-Operationen auf verschiedene Zählervariablen.

Neben diesen reinen Synchronisationsmechanismen besitzen auch die im nächsten Kapitel behandelten Mechanismen zum Informationsaustausch zwischen Prozessen ohne gemeinsamen Datenbereich die Eigenschaft, daß mit ihnen die Abläufe der beteiligten Prozesse koordiniert werden.

4.1.3 Interprozeßkommunikation

Für den Informationsaustausch zwischen Prozessen (*inter process communication*, IPC) werden von Betriebssystemen verschiedene Objekte angeboten [30]. Die wichtigsten sind:

- Dateien,
- Pipes,
- Message Queues,
- Shared Memory-Elemente und
- Sockets.

Die Kommunikation über Dateien, d.h. ein Prozeß schreibt Informationen in eine Datei, die ein anderer Prozeß liest, ist in Bild 4.3 verdeutlicht. Der Programmierer hat Vorkehrungen zu treffen, daß die Reihenfolge der Abläufe eingehalten wird sowie der Dateiname eindeutig und den beteiligten Instanzen bekannt ist. Die Daten bleiben, sofern sie nicht explizit gelöscht werden, dauerhaft gespeichert. Sie sind auch für die an der Kommunikation nicht direkt beteiligten Prozesse verfügbar. Das Betriebssystem verhindert zwar gleichzeitige Schreibzugriffe, nicht jedoch Schreib/Lese-Inkonsistenzen.

Pipes und Message Queues stellen sequentielle Kommunikationskanäle dar, über die nach dem FIFO-Prinzip Informationen transferiert werden. Wie in Bild 4.4 dargestellt, sind sie gerichtet, können eine begrenzte Informationsmenge zwischenspeichern und sind somit mit einer Warteschlange vergleichbar. Die Sende- und Empfangsvorgänge der verschiedenen Prozesse werden über den Kommunikationskanal synchronisiert.

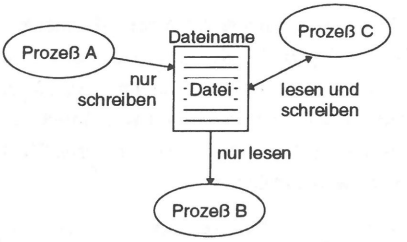


Bild 4.3: Interprozeßkommunikation über Dateien

Pipes arbeiten zeichenweise und können aus der Sicht der Prozesse wie sequentielle Dateien behandelt werden, die entweder nur zum Lesen oder nur zum Schreiben geöffnet sind. Message Queues entsprechen Meldungswarteschlangen im Hauptspeicher und besitzen, wie die Prozesse selbst, eine vom Betriebssystem vergebene, nur für die Dauer ihrer Existenz eindeutige Systemkennung. Globalen Message Queues wird zusätzlich ein systemweit eindeutiger *Key-Value* zugeordnet, der mit einem Dateinamen vergleichbar ist und über den die aktuelle Systemkennung ermittelt werden kann.

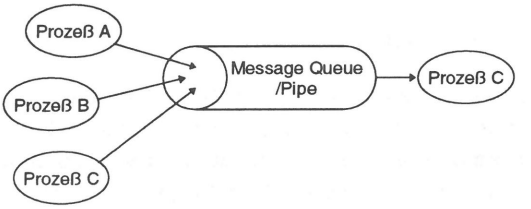


Bild 4.4: Interprozeßkommunikation über Pipes/Message Queues

Aus einer Pipe oder einer Message Queue kann jede Information nur einmal ausgelesen werden. Sind keine Daten vorhanden, blockiert das Betriebssystem den lesenden Prozeß. Ist eine Warteschlange voll, wird der sendende Prozeß blockiert. Über diese Blockierungen werden Sender und Empfänger synchronisiert. Ist dies nicht erwünscht, kann auf blockierungsfreie Zugriffsfunktionen zurückgegriffen werden, die den Fehlschlag der Operation über einen Fehlercode anzeigen. In der Regel wird jedem Prozeß exklusiv ein Kommunikationskanal zugeordnet, in den alle für den betreffenden Prozeß bestimmte Daten gelangen, so daß der Sender den Kommunikationskanal und damit die Instanz des Zielprozesses kennen muß.

Ein Shared Memory-Element (SHM) erlaubt, wie in Bild 4.5 dargestellt, Prozessen mit getrennten Datenbereichen, sich einen bestimmten Speicherbereich zu teilen. Einmal von einem Prozeß angelegt, können andere Prozesse dieses Element in ihren Adressbereich einblenden und somit quasi gleichzeitig darauf zugreifen, wobei keinerlei Synchronisation durch das Be-

triebssystem erfolgt. Wie Message Queues besitzen SHM-Elemente eine Systemkennung und einen optionalen *Key-Value*. Analog zu der Kommunikation über Dateien benötigt ein Prozeß keinerlei Informationen über die anderen an der Kooperation beteiligten Prozesse. Zu beachten ist, daß die Adresse, an die ein SHM-Bereich im lokalen Adreßraum eines Prozesses eingeblendet wird, variieren kann und deshalb Zeiger innerhalb eines SHM-Elements immer relativ zur Anfangsadresse angegeben werden müssen.

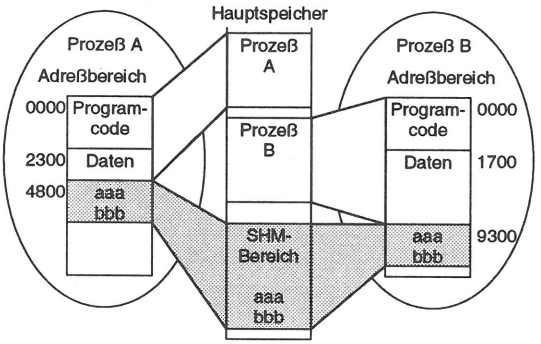


Bild 4.5: Interprozeßkommunikation über ein Shared Memory-Element

Sockets stellen Kommunikationsendpunkte eines Prozesses dar, die es erlauben, Daten an andere Prozesse zu übermitteln, ohne sich über den Übertragungsweg Gedanken machen zu müssen. Die kommunizierenden Prozesse können sich dabei auf unterschiedlichen, untereinander vernetzten Rechnersystemen befinden. Sockets bilden eine allgemeine Anwendungsschnittstelle für verschiedene Transportdienste und Protokolle, wobei die TCP/IP-Protokolle vorherrschen [18]. Der Datenaustausch über Sockets innerhalb eines Rechnersystems erfolgt über Pipes. Der Socket-Mechanismus stammt ursprünglich aus der Berkeley-Variante des UNIX-Betriebssystems [134], ist mittlerweile aber auch in anderen Betriebssystemen zu finden.

4.1.4 Mehrprozeßsysteme

Die Hauptschwierigkeiten in der Entwicklung von Mehrprozeßsystemen liegen in der Koordinierung der Programmabläufe und des Informationsaustausches. Diese Schwierigkeiten beruhen auf:

- der Nichtvorhersehbarkeit der Zuteilung der Rechenzeit an die Prozesse,
- der Möglichkeit der gegenseitigen Prozeßblockierung (Verklemmung, deadlock),
- der Gefahr eines unproduktiven, endlosen Prozeßwechsels (lifelock),
- der möglichen Veränderung der Meldungsreihenfolge und
- der Erzeugung und Interpretation der Meldungen zwischen Prozessen.

Demgegenüber stehen folgende Vorteile eines Mehrprozeßsystems im Vergleich mit einem monolithischen Programm:

- einfacherer Programmablauf der einzelnen Prozesse, verbunden mit einer geringeren Fehlerträchtigkeit,
- bessere Modellierbarkeit paralleler Aktivitäten,
- Isolation von Fehlfunktionen auf einen Prozeß,
- starke Kapselung der Daten und des Programmcodes,
- genau definierte Schnittstellen zwischen den Prozessen und
- größere Stabilität des Gesamtsystems.

Bei entsprechendem Design ist es sogar möglich, zur Laufzeit die Programme für einzelne, momentan nicht aktive Prozesse auszutauschen, wodurch Funktionserweiterungen während des Betriebs möglich werden. Prinzipiell ist die Effizienz eines Mehrprozeßsystems aufgrund des zusätzlichen Aufwandes für die Prozeßwechsel und die Interprozeßkommunikation etwas schlechter als bei einer monolithischen Implementierung. Das Betriebssystem ist jedoch in der Lage, während der Durchführung von Ein/Ausgabeoperationen andere Prozesse zu bearbeiten und somit diese Zeiten zu nutzen. Für den Benutzer ergibt sich im allgemeinen ein besseres Zeitverhalten, da interaktive Vorgänge von relativ kurzer Dauer Hintergrundaktivitäten mit langen Rechen- oder Wartezeiten vorgezogen werden können, was zu einer höheren Akzeptanz führt.

4.1.5 Mehrprozeßsysteme unter UNIX

Das Betriebssystem, unter dem ein Mehrprozeßsystem arbeitet, hat einen starken Einfluß auf dessen Laufzeiteigenschaften und muß deshalb mit in die Entwicklung der Prozeßstruktur einbezogen werden. Wichtig in diesem Zusammenhang sind:

- die Prozeßzuteilungsstrategie,
- die verfügbaren IPC-Mechanismen und
- die Rechenzeiten für Prozeßwechsel, Prozeßerzeugung und Interprozeßkommunikation.

Auf Workstations, der Zielplattform des Monitoring-Systems, ist das Betriebssystem UNIX weit verbreitet. Die dort verwendete Prozeßzuteilungsstrategie arbeitet nach dem *Round Robin-Prinzip* mit mehreren Prioritätsebenen und Rückkopplungen [4]. Die Prioritätsebenen, dargestellt in Bild 4.6, umfassen mehrere Prozeßprioritäten, die sich gegenseitig unterbrechen. Auf der Ebene der Echtzeitprozesse wird für die Prozessorzuteilung ein reines Zeitscheibenverfahren mit festen Prioritäten verwendet. Die übrigen Prozesse fallen in die Time-Sharing-Klasse, aus der sie nur während der Ausführung von Systemfunktionen in die Systemklasse wechseln.

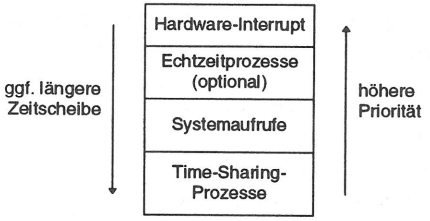


Bild 4.6: Prioritätsklassen unter UNIX

Der Zuteilungsalgorithmus für Prozesse der Time-Sharing-Klasse, aus denen ein Mehrprozeßsystem besteht, ist recht kompliziert und unterscheidet sich in Details bei unterschiedlichen UNIX-Derivaten [4], [131]. Allen gemeinsam ist das Ziel, eine gute Prozessorauslastung bei zügigem interaktivem Arbeiten zu erreichen. Darüber hinaus wird versucht, die Rechenzeiten möglichst gerecht zwischen den Anwendungen aufzuteilen. Dies wird dadurch erreicht, daß ein Prozeß während seiner Laufzeit, abhängig davon, wieviel Rechenzeit er in der letzten Vergangenheit in Anspruch genommen hat, regelmäßig eine neue Ausführungspriorität zugewiesen bekommt. Erhielt ein Prozeß viel Rechenzeit, sinkt seine Priorität, war er jedoch inaktiv oder mußte er auf die Ausführung warten, erhöht sie sich entsprechend. So wird sich ein Batch-Prozeß auf einer niedrigen Prioritätsstufe einpendeln, während ein interaktiver Prozeß, wie z.B. eine Editoranwendung, eine hohe Stufe beibehält und somit schnell reagieren kann. Die Neuberechnung der Ausführungsprioritäten aller ablauffähigen Prozesse erfolgt periodisch, während der eigentliche Zuteilungsvorgang und damit die Prozeßwechsel jedesmal stattfinden, nachdem der Prozessor aus dem Systemmodus (*Kernel Mode*) in den Anwendungsmodus (*User Mode*) wechselt, bzw. eine Zeitscheibe endet.

Die Ausführungspriorität eines Prozesses hängt auch von einer einstellbaren Basispriorität ab, über die der Anwender oder Programmierer die Aufteilung der Rechenzeit unter den Prozessen beeinflussen kann. Es ist zu beachten, daß auch Anwendungen mit niedriger Basispriorität in Hochlastphasen regelmäßig eine Zeitscheibe zugeteilt wird und sie somit höherpriore Prozesse behindern (Prioritätsumkehr). Mit niedrigerer Ausführungspriorität steigt die Größe der zur Verfügung stehenden Zeitscheibe, so daß mehrere parallele Batch-Programme (niedrige Prioritätsstufe) aufgrund der geringeren Prozeßwechselrate effizienter ablaufen.

Echtzeitbetriebssysteme garantieren die Einhaltung maximaler Verzögerungszeiten auf externe Ereignisse und besitzen Prozessorzuteilungsstrategien, die es erlauben, zeitkritische Prozesse innerhalb ihrer Zeitschranken zu bearbeiten. Dabei muß jedoch sichergestellt sein, daß die Kapazitätsgrenze durch die zeitkritischen Prozesse nicht überschritten wird. Niederpriore Prozesse dürfen darauf jedoch keinen Einfluß haben. Herkömmliche UNIX-Derivate besitzen diese Eigenschaft nicht. Die Echtzeit-Prioritätsstufe setzt nämlich voraus, daß alle Systemaufrufe wiedereintrittsfähig (*re-entrant*) und somit unterbrechbar sind, die maximale, nicht unterbrech-

bare Ausführungsdauer eines Systemaufrufs beschränkt ist und Interrupt-Routinen eine definierte maximale Bearbeitungsdauer besitzen. Dies ist, außer bei speziellen echtzeitfähigen Derivaten, nicht der Fall.

Trotzdem kann es für echtzeitabhängige Anwendungen, bei denen eine Verletzung der Echtzeitbedingungen keine fatalen Auswirkungen hat, durchaus sinnvoll sein, UNIX als Laufzeitumgebung einzusetzen, da es auch für Hardwareplattformen verfügbar ist, deren Leistungsfähigkeit und Preis/Leistungsverhältnis Echtzeitsysteme um ein Vielfaches übertreffen. Voraussetzung für den Einsatz eines echtzeitabhängigen Systems unter UNIX ist jedoch, daß die Bearbeitungszeiten der einzelnen Aktivitäten genügend kurz sind und genügend Leistungsreserve bezüglich der Rechenkapazität existiert. Ebenso ist der Hauptspeicher so zu dimensionieren, daß die Prozesse der echtzeitabhängigen Anwendung nicht ausgelagert werden müssen.

Moderne UNIX-Varianten besitzen alle in Kapitel 4.1.3 vorgestellten IPC-Mechanismen. Zur Ermittlung ihrer Leistungskenngrößen wurden vergleichende Messungen der Übertragungszeiten von langen und kurzen Meldungen auf der Zielplattform des Monitoring-Systems durchgeführt [133]. Für die Zeiterfassung standen lediglich die recht ungenauen, vom System ausgegebenen Rechenzeiten und Realzeiten zur Verfügung, so daß die IPC-Zugriffe mehrfach wiederholt und die Gesamtlaufzeit gemessen wurde. Tabelle 4.1 enthält die mittleren Rechenzeiten für einen Meldungstransfer, bestehend aus einem Sende- und einem Empfangsvorgang ohne den erforderlichen Prozeßwechsel, aufgeschlüsselt nach der Meldungslänge und den verschiedenen IPC-Mechanismen. Die Messungen wurden unter dem UNIX-Betriebssystem der Firma Santa Cruz Operation Inc. (SCO) auf einem Personal Computer mit einem Prozessor des Typs INTEL 486-DX2, dessen interne Taktfrequenz 66 MHz betrug, durchgeführt. Die Werte sind aufgrund der starken Streuung der Einzelmessungen nur als grobe Anhaltswerte zu verstehen.

	Datei	Pipe	nur SHM	SHM + Semaph.	Message Queue	Socket
Kurze Meldung (1 Byte)	200 µs	150 µs	0,1 µs	60 µs[1]	140 µs	500 µs (200 ms)
Lange Meldung (1200 Bytes)	370 µs	320 µs	56 µs[2]	116 µs	290 µs	1800 µs (200 ms)

Tabelle 4.1: Übertragungszeiten bei verschiedenen IPC-Mechanismen (unidirektional)

[1] Realisierung einer einseitigen Synchronisation mit zwei Semaphor-Operationen mit je 30 µs.

[2] Unter Verwendung der Systemfunktion *memcpy()*.

Die bei Message Queues benötigte Rechenzeit für den Meldungstransfer setzt sich aus einem konstanten Anteil für die Queue-Verwaltung und den Aufruf der Betriebssystemfunktionen, sowie einem vom Datenvolumen abhängigen Anteil, hervorgerufen durch den Kopiervorgang zwischen dem lokalen Speicherbereich und dem der Message Queue, zusammen. Die Kommunikation über Pipes stellte sich als nicht wesentlich langsamer heraus, da der Datentransfer allein im Hauptspeicher stattfand. Lediglich das Anlegen, Öffnen, Schließen und Löschen einer Pipe dauerte länger. Dateien sind bei kurzen Meldungen mit Pipes und Message Queues vergleichbar, da auch hier wiederum der Transfer im Hauptspeicher stattfindet. Erst bei Meldungsgrößen über 60 KByte traten die Ein/Ausgabeoperationen mit Laufzeiten im Millisekundenbereich in den Vordergrund. Der Zugriff auf den Speicherbereich von Shared Memory-Elementen unterscheidet sich nicht von dem auf lokale Daten und ist somit am schnellsten. Es ist jedoch der Synchronisations-Overhead durch Semaphore zu berücksichtigen. Insgesamt ist dies die schnellste Art der Interprozeßkommunikation, wenn eine Speicherung mehrerer Meldungen zwischen den Prozessen nicht erforderlich ist.

Die Socket-Messungen erfolgten lokal auf einem Rechner, jedoch unter der Verwendung des TCP/IP-Protokolls. Diese Kommunikation ist zwar universell, aber relativ langsam. Bei zeichenorientierter Übertragung findet im Socket eine Gruppierung in 1 KByte-Blöcke statt, die viele kurze Pakete auf dem Netz verhindern soll [124]. Dies führt jedoch dazu, daß die Absendung eines nicht gefüllten Pakets verzögert wird, wodurch sich die in Klammern angegebenen realen Verzögerungszeiten ergeben. Die anderen Zahlen wurden hochgerechnet, aus Messungen mit 1024 Byte Meldungslänge (1,6 ms) und 2048 Bytes Meldungslänge (2,7 ms).

4.2 Architektur des Monitoring-Systems

Nach der Vorstellung der prinzipiellen Architektur des Monitoring-Systems und den Möglichkeiten der Einordnung in ein globales Netzmanagementkonzept in Kapitel 3 soll hier der Kern des Systems, die Meßauftragsabwicklung, beschrieben werden.

4.2.1 Grundkonzept

Das Monitoring-System verwaltet alle Messungen als dynamisch angelegte Datenstrukturen, den *Meßjobs*, in denen die Parameter und Ergebnisse der Meßaufträge abgelegt sind. Der Meßjob ist als Shared Memory-Element ausgeführt und repräsentiert den Meßauftrag im System. Ein Meßjob ist, wie in Bild 4.7 dargestellt, in fünf Bereiche unterteilt. Der Meßauftragskopf enthält, neben Angaben zum Meßauftragstyp und dem zeitlichen Ablauf der Messung, Referenzen und Längenangaben zu den übrigen Bereichen, deren Feinstrukturen vom Typ des Meßauftrags abhängig sind.

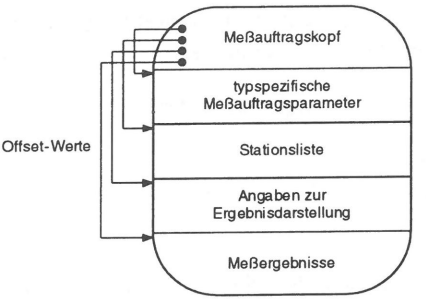


Bild 4.7: Struktur des Meßjobs

In einem globalen Shared Memory-Element befindet sich die *Jobliste*, mit deren Hilfe alle Meßaufträge im System verwaltet werden. Sie besitzt einen Kopf, welcher die Belegungszustände der Listenelemente und die Systemkennungen der globalen Prozesse enthält. Jeder Meßauftrag belegt ein Element in der Jobliste. Das Joblistenelement beinhaltet die Systemkennung des Meßjobs, die Kennungen der an der Messung beteiligten Prozesse sowie den Startzeitpunkt und den Ausführungszustand des Meßauftrags. Über die Jobliste ist ein Zugriff auf die Parameter, Ergebnisse und Verwaltungsinformationen möglich. Letzteres ist hilfreich, falls ein Meßauftrag abgebrochen werden soll, sei es, daß der Anwender die Messung nicht mehr benötigt oder der Meßauftrag aufgrund eines Fehlers nicht korrekt beendet werden kann. Bild 4.8 zeigt die Shared Memory-Elemente des Monitoring-Systems mit ihren Beziehungen untereinander.

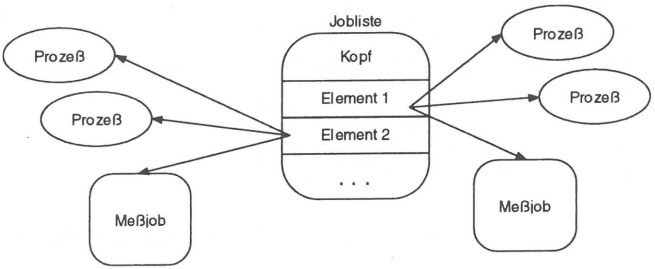


Bild 4.8: Shared Memory-Elemente des Monitoring-Systems

Der Zugriff auf die Jobliste wird durch das *Joblisten-Semaphor* kontrolliert. Eine Trennung zwischen exklusivem Schreib/Leserecht und einem mehrfachen Nur-Lesezugriff wurde unterlassen, da die Zugriffe auf die Jobliste generell von kurzer Dauer sind und reine Lesezugriffe selten vorkommen. Für den Zugriff auf die Meßjobs wurden keine Semaphore vorgesehen, da

durch den Ablauf einer Messung konkurrierende Schreibzugriffe ausgeschlossen sind und Schreib/Lesekonflikte effizienter durch Schloßvariablen aufgelöst werden können.

Ein Meßauftrag kann in drei Abschnitte gegliedert werden:

- die Auftragserzeugung,
- die Meßdurchführung und
- die Ergebnisdarstellung.

Jeder dieser Abschnitte wird von einem eigenen Prozeß bearbeitet, so daß jede Messung einen Eintrag in der Jobliste, einen Meßjob und drei zugeordnete Prozesse umfaßt. Bild 4.9 zeigt die interne Struktur der Meßauftragsabwicklung. Die Flexibilität des Systems ergibt sich aus den Kombinationsmöglichkeiten der Programme für die verschiedenen Prozesse, welche über die Jobliste und den Meßjob zusammengehalten werden.

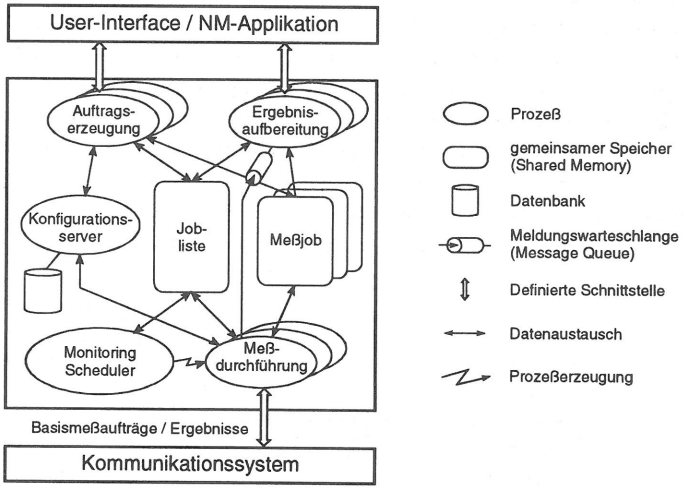


Bild 4.9: Die Meßauftragsabwicklung des Monitoring-Systems

Die *Auftragserzeugung (AEZ)* interpretiert die Meßauftragsbeschreibung (vgl. Kapitel 3.4) und prüft, ob die gewünschte Messung mit den angegebenen Meßkomponenten durchführbar ist. Ist dies der Fall, wählt sie geeignete Programme für die Meßdurchführung und die Ergebnisaufbereitung aus und erzeugt einen neuen Meßjob. Schließlich werden die benötigten Parameter aus der Meßauftragsbeschreibung extrahiert und im Meßjob abgelegt. Die von der AEZ definierten Parameter eines Meßjobs sind für die gesamte Lebensdauer des Meßauftrags unveränderlich.

Der Zugriff auf die Netz- und Meßsystemkonfiguration, deren Struktur in Kapitel 3.5 vorgestellt wurde, erfolgt über einen *Konfigurationsserver*, der in Kapitel 4.2.3 genauer beschrieben ist. Der Konfigurationsserver übernimmt die Zuordnung von Stationsnamen zu Stationsadressen und umgekehrt sowie die Überprüfung, ob mit der angegebenen Komponente eine bestimmte Messung überhaupt durchführbar ist. Oft sind die Meßkomponenten nicht in der Lage, mehrere Messungen gleichzeitig auszuführen, so daß es zu Belegungskollisionen bei den Meßkomponenten kommen kann. Als zentrale Instanz ist es dem Konfigurationsserver möglich, über die Meßaufträge Buch zu führen und neue Meßaufträge abzuweisen, die mit bestehenden Meßaufträgen kollidieren würden (siehe Kapitel 4.2.4).

Damit Messungen zu einem späteren Zeitpunkt eingeplant werden können, bedarf es eines zentralen Prozesses, dem *Monitoring Scheduler*, der die Jobliste überwacht, alle neuen Meßaufträge registriert und zum angegebenen Zeitpunkt einen Prozeß für die Meßdurchführung erzeugt. Dieser Prozeß, auch Meßprozeß genannt, führt ein in der Jobliste angegebenes Meßprogramm aus. Damit der Monitoring Scheduler für das Erkennen neuer Meßaufträge keine Rechenzeit in Anspruch nehmen muß, ist der Jobliste ein weiteres Semaphor zugeordnet, das von der Auftragserzeugung für jeden neu generierten Meßauftrag inkrementiert wird. Der Monitoring Scheduler dekrementiert das Semaphor entsprechend und sucht im Erfolgsfall das zugehörige Element der Jobliste. Abhängig von der angegeben Startzeit wird der Meßprozeß sofort erzeugt oder es wird ein Timer gesetzt, der den Scheduler zu gegebener Zeit an den Meßauftrag erinnert. Liegt kein neuer Meßauftrag vor, wartet der Scheduler an dem Semaphor, bis ein neuer Meßauftrag eintrifft oder ihm durch ein Signal des Timers mitgeteilt wird, daß der Zeitpunkt gekommen ist, einen bereits registrierten Meßauftrag zu starten.

Die *Meßdurchführung* ist für die Ausführung der Messung gemäß den Angaben im Meßjob verantwortlich. Die Integration unterschiedlicher Meßsysteme und Kommunikationsprotokolle erfolgt durch die entsprechenden Meßprogramme, die zusammen mit den Meßkomponenten in der Lage sind, eine bestimmte Klasse von Meßaufträgen durchzuführen. Im Prinzip lassen sich so alle Meßkomponenten, die auf irgendeine Weise fernsteuerbar sind, einheitlich über das Monitoring-System bedienen. Bei der Ausführung eines Meßauftrags wird dieser in Basismeßaufträge aufgespalten, die von den Meßkomponenten durchgeführt werden. Die Ergebnisse dieser Basismeßaufträge werden dann zu Zwischen- oder Endergebnissen des Meßauftrags zusammengesetzt, die im Ergebnisteil des Meßjobs abgelegt werden. Zwischenergebnisse, die sofort angezeigt werden sollen, schickt der Meßprozeß als Meldungen an die Message Queue der Ergebnisaufbereitung, deren Systemkennung aus dem Joblistenelement entnommen wird. Hierzu ist es notwendig, daß die Ergebnisaufbereitung zum Zeitpunkt, an dem das erste Zwischenergebnis anfällt, bereits existiert und die Kennung ihrer Message Queue in die Jobliste eingetragen hat. Die Speichereigenschaft der Message Queue sichert die richtige Reihenfolge der Zwischenergebnisse und schützt vor Informationsverlusten, falls die Ergebnisaufbereitung ihren Aufgaben kurzfristig nicht nachkommt.

Die Ergebnisse, die eine Meßdurchführung generiert, umfassen alle Daten, die aus der Messung gewonnen werden können. Lediglich bei periodischen Messungen ist eine statistische Zusammenfassung zur Begrenzung des Datenaufkommens vorgesehen. Die Anpassung an die Bedürfnisse des Anwenders erfolgt durch die *Ergebnisaufbereitung (EAB)*. Besteht der Anwender aus einer Netzmanagementapplikation, wird die EAB so ausgelegt, daß die erwarteten Informationen im richtigen Format zur Verfügung gestellt werden. Ein menschlicher Benutzer kann Detaillierungsgrad und die Art der Präsentation, welche weitgehend unabhängig von der Meßfunktion sind und sich am Ziel der Messung orientieren, bei der Erzeugung des Meßauftrags mit angeben. Dadurch kann dieselbe Meßfunktion und damit dasselbe Meßprogramm für eine Hintergrundmessung zum Erstellen einer Langzeitstatistik, eine Überwachungsmessung, die gegebenenfalls einen Alarm erzeugt, oder für eine spontane Messung zum Aufspüren einer Fehlfunktion im Netz verwendet werden. Hierzu werden Programme mit entsprechenden Aufbereitungsfunktionen für die verschiedenen Meßauftragsklassen als Basis für die EAB-Prozesse bereitgestellt. Die Meßauftragsbeschreibung definiert zu einer Messung genau eine EAB, es können jedoch auch andere EAB-Prozesse auf den Meßjob einer laufenden Messung zugreifen. Somit kann ein Meßauftrag für mehrere Überwachungsaktivitäten verwendet werden, oder es ist möglich, die Ergebnisse mehrerer Meßaufträge zu kombinieren.

Die Darstellung der Meßergebnisse für den menschlichen Anwender erfolgt durch das *User Interface* (UIF), dessen Schnittstelle zu der Ergebnisaufbereitung unabhängig davon ist, ob eine grafische oder eine zeichenorientierte Benutzungsoberfläche vorliegt. Auf diese Schnittstelle wird in Kapitel 4.2.5 näher eingegangen. Das User Interface besitzt eine weitere Schnittstelle für die Erzeugung von Meßaufträgen, die als Programmaufruf der Auftragserzeugung realisiert ist, dessen Parameter die komplette Meßauftragsbeschreibung beinhaltet. Aufgabe des User Interfaces ist es, die anwendergerechte Bedienung des Monitoring-Systems zu ermöglichen. Hierzu gehören auch Funktionen zur Verwaltung des Gesamtsystems, wie die Anzeige des Inhalts der Jobliste, der Meßjobs und die Anzeige und Verwaltung von Fehlermeldungen der verschiedenen Prozesse.

Die verschiedenen Überwachungsaufgaben erfordern unterschiedliche Arten der Ergebnisdarstellung. Diese sind:

- die Anzeige zusammengefaßter Ergebnisse nach der Messung,
- die Anzeige aller Zwischenergebnisse nach der Messung,
- die laufende Anzeige der Zwischenergebnisse während der Messung,
- die Anzeige des aktuellen Zustandes während der Messung auf Anfrage und
- der kontinuierliche Vergleich mit Grenzwerten und ggf. eine Alarmauslösung.

Aus diesen Anforderungen ergibt sich der Zeitpunkt, an dem der Prozeß für die Ergebnisaufbereitung gestartet werden muß, und die Art, wie der Meßprozeß Zwischenergebnisse zu übertragen hat. Im Monitoring-System sind vier Zeitpunkte für den Start der Ergebnis-

aufbereitung vorgesehen, die in der Meßauftragsbeschreibung wie folgt angegeben werden können:

- At_Once: zu Beginn der Messung (zur laufenden Anzeige der Zwischenergebnisse),
- When_Done: automatisch nach Beendigung der Messung,
- By_Shell: Start durch den Anwender zu einem beliebigen Zeitpunkt,
- Proc_Exists: der EAB-Prozeß existiert bereits.

Der letzte Fall, daß der EAB-Prozeß bereits existiert, kann für den Ablauf der Messung wie der Fall At_Once behandelt werden. Lediglich der Monitoring Scheduler muß die Erzeugung des Prozesses überspringen. Die Übertragung der Ergebnisse zwischen Meßprozeß und Ergebnisaufbereitung kann über den Meßjob oder die Message Queue der EAB erfolgen. Die in Kapitel 3.4 beschriebenen Modi zur Ergebnisdarstellung definieren, wie dies stattzufinden hat. Der Modus Queue_And_Save ermöglicht der Ergebnisaufbereitung, Zwischenergebnisse sofort anzuzeigen, gleichzeitig aber stehen die Daten auch anderen Prozessen im Meßjob zur Verfügung. Das Abspeichern aller Zwischenergebnisse (Save_All) kann unter Umständen einen hohen Speicherbereich im Meßjob erfordern. Dieser Modus verbietet sich für Dauermessungen, da die Größe des Meßjobs beim Anlegen bekannt sein muß. Bei Langzeitmessungen empfiehlt es sich deshalb, die Zwischenergebnisse überschreibend abzulegen (Overwrite) oder als Meldungen zu verschicken (Queue) und die Informationen in der Ergebnisaufbereitung bei Bedarf in einer Datei zu sichern. Tabelle 4.2 zeigt die Auswertemöglichkeiten für die verschiedenen Kombinationen zwischen dem Ausgabemodus und dem Start der Ergebnisaufbereitung.

	At_Once	When_Done	By_Shell
Save_All	Abfrage aller Ergebnisse zu jeder Zeit durch den Benutzer	Analyse aller Ergebnisse nach der Messung	Benutzer kann frei auf die Meßergebnisse zugreifen
Overwrite	Abfrage des letzten Ergebnisses zu jeder Zeit durch den Benutzer	* nicht anwendbar * (Zwischenergebnisse gehen verloren)	beliebiger Zugriff auf aktuelle Werte (sinnvoll für Dauermessungen)
Queue	Automatische Aktualisierung durch neue Zwischenergebnisse	* nicht anwendbar *	* nicht anwendbar *
Queue_And_Save	Meßjob auch für andere Ergebnisaufbereitungen verfügbar	* nicht anwendbar *	* nicht anwendbar *

Tabelle 4.2 Mögliche Ausführungsarten einer Messung mit Auswertemöglichkeiten

Durch die Zuordnung der Ablaufphasen einer Messung zu einzelnen Prozessen wird eine Flexibilität auf verschiedenen Ebenen erreicht. Zum einen ermöglicht es eine zeitliche Überlappung zwischen der Meßdurchführung und der Ergebnisdarstellung, ohne daß dies bei der Implementierung berücksichtigt werden muß. Das Monitoring-System ist zudem offen für neue Meßkomponenten durch die Möglichkeit der Einbindung entsprechender Meßprogramme, ohne den Überwachungsbetrieb zu stören. Schließlich kann durch die Trennung zwischen Meßprozeß und Ergebnisaufbereitung die Art der Darstellung unabhängig vom Meßauftrag an die Überwachungsanforderungen angepaßt werden. Die individuelle Zuordnung der Softwarekomponenten zu den Meßaufträgen erlaubt während des Überwachungsbetriebs eine Erweiterung um neue Meßauftragsklassen, indem entsprechende Strukturen für den Meßjob definiert, Programme für die Meßdurchführung und die Ergebnisaufbereitung zur Verfügung gestellt und die Auftragserzeugung um die neuen Meßaufträge erweitert werden.

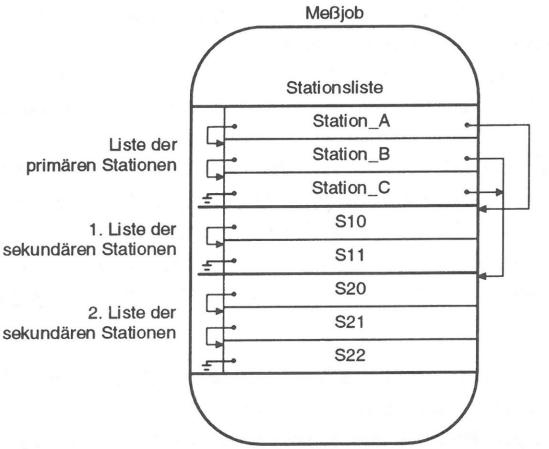
4.2.2 Die Umsetzung der Meßauftragsbeschreibung

Die Auftragserzeugung besteht im wesentlichen aus einem Parser für die Meßauftragsbeschreibungssprache, der während der Analyse eines Meßauftrags dynamisch eine hochgradig verzeigerte Zwischendatenstruktur (ZDS) erzeugt [3]. Diese Struktur stellt einen Parameterbaum dar, der zum Schluß alle Informationen des Auftrags enthält. Für fehlende Parameter werden, soweit dies unabhängig vom Meßauftrag möglich ist, Defaultwerte eingesetzt. Bei an sich redundanten Parametern werden diese aus den übrigen Angaben berechnet. In einer nachfolgenden semantischen Analyse wird, abhängig vom Typ des Meßauftrags, geprüft, ob alle Meßparameter zulässig sind und innerhalb definierter Wertebereiche liegen. Dabei werden auch meßauftragsspezifische Defaultwerte in die ZDS eingetragen. Danach werden die Größe des SHM-Elements für den Meßjob bestimmt, das Element selbst erzeugt und die Auftragsparameter übertragen. Der Meßjob besitzt eine relativ einfache Datenstruktur unter Vermeidung von Referenzen innerhalb des Shared Memory-Elements, damit die Meßdurchführung bzw. die Ergebnisaufbereitung einfach darauf zugreifen können.

Die Spezifikation der Meßprogramme und Programme für die Ergebnisaufbereitung erfolgt durch die Angabe des Programmnamens und des Ortes im Dateisystem in den komplexen Parametern MEAS[] und OUTPUT[]. Existiert der Prozeß der Ergebnisaufbereitung bereits, muß statt des Dateinamens die Systemkennung des Prozesses angegeben werden. Der Parameter OUTPUT[] legt zusätzlich fest:

- den Startzeitpunkt der EAB,
- den Übertragungsmodus der Meßergebnisse und
- zusätzliche Ausführungsparameter für die EAB.

Der Startzeitpunkt der EAB kann häufig aus deren Spezifikation abgeleitet werden. Ist z.B. keine Ergebnisaufbereitung definiert, ergibt sich dieser zu By_Shell (Start durch den Benutzer); wird statt des Dateinamens eine Systemkennung angegeben, kommt nur Proc_Exists in Frage. Startzeitpunkt und Übertragungsmodus sind voneinander abhängig (siehe Tabelle 4.2), wodurch sich manche Kombinationen verbieten. Dies ist in der Syntax des im Anhang vollständig spezifizierten Output-Parameters berücksichtigt.



STAT[Station_A -> (S10, S11), (Station_B, Station_C) -> (S20, S21, S22)]

Bild 4.10: Umsetzung einer Stationsbeschreibung in die Stationsliste des Meßjobs

Die Angaben des komplexen Parameters STAT[], der bereits in Kapitel 3.4 beschrieben wurde, werden bei der Auftragserzeugung in die Stationsliste des Meßjobs übertragen. Diese Liste enthält die primären Stationen, wobei jeder primäre Stationseintrag, wie in Bild 4.10 dargestellt, auf eine Liste mit sekundären Stationen verweisen kann. Bei der Angabe von Stationsgruppen werden diese mit Hilfe der Konfigurationsdatenbank in mehrere Stationseinträge aufgelöst. Beziehen sich mehrere primäre Stationen auf dieselbe Menge von sekundären Stationen, wird die Liste der sekundären Stationen nur einmal im Meßjob abgelegt und mehrfach darauf verwiesen.

4.2.3 Verwaltung der Konfigurationsdaten

Die Speicherung und Bereitstellung der Konfigurationsinformationen für das Monitoring-System, deren Struktur in Kapitel 3.5 vorgestellt wurde, erfolgt entweder zusammen mit den Konfigurationsdaten des übrigen Managements in der dortigen Managerdatenbank oder muß

vom Monitoring-System selbst übernommen werden. Auf jeden Fall bietet sich der Einsatz eines relationalen Datenbanksystems an, da es folgende Vorteile besitzt:

- ein Entity-Relationship-Diagramm kann vollständig in ein relationales Datenmodell umgesetzt werden,
- das System vereinfacht den Aufbau der komplexen Datenstrukturen und Beziehungen,
- Eingabe, Ausgabe und Pflege der Daten erfolgen außerhalb des Monitoring-Systems,
- es existiert eine standardisierte Abfragesprache (SQL, Structured Query Language), die auch aus Programmen heraus anwendbar ist,
- die Datenstruktur kann geändert werden, ohne existierende Informationen zu verlieren,
- das Datenbanksystem gewährleistet eine hohe Datensicherheit und Konsistenz.

Leider hat die hohe Flexibilität und Sicherheit der relationalen Datenbanksysteme auch ihren Preis. Im Rahmen der Prototypentwicklung des Monitoring-Systems zeigte es sich, daß der Aufbau einer Verbindung zwischen einer Applikation und dem Datenbanksystem eine inakzeptable Dauer von drei bis acht Sekunden betrug. Einzelne Abfragen waren zwar etwas schneller (Echtzeit 0,5 - 1 Sekunde, Systemzeit 40 - 60 ms), jedoch für häufige Abfragen immer noch zu langsam.

Das Ziel, kurze Antwortzeiten für häufige Abfragen unter Erhaltung der Flexibilität eines Datenbanksystems zu erreichen, ist möglich durch den Einsatz eines zentralen Konfigurationsservers, der alle Anfragen bezüglich der Konfiguration des Netzes und der Meßstationen bearbeitet. Dadurch muß sich nur ein Prozeß während der Initialisierung des Monitoring-Systems beim Datenbanksystem anmelden, und es besteht die Möglichkeit, häufig benötigte Informationen über das Meßsystem und stark dynamische Statusinformationen lokal zwischenzuspeichern. Die Kommunikation mit den übrigen Prozessen erfolgt über Message Queues, da es sich bei dem Konfigurationsserver um einen zentralen Prozeß handelt, bei dem Anfragen bis zu ihrer Bearbeitung gespeichert werden müssen. Aufgrund der sequentiellen Abarbeitung der Operationen auf die Konfigurationsdaten treten keine Konsistenzprobleme auf.

Bild 4.11 zeigt das Architekturmodell des Konfigurationsservers. Die Zwischenspeicherung der häufig benötigen Informationen erfolgt in Binärfiles, deren Inhalt bei der Initialisierung aus dem Datenbanksystem gewonnen werden kann und bei Modifikationen zurückgeschrieben wird. Für Testzwecke und zur Schaffung der Möglichkeit, den Konfigurationsserver auch ohne Datenbanksystem einsetzen zu können, erlauben Textfiles dem Anwender, die Daten für die Binärfiles in lesbarer Form einzugeben bzw. zu betrachten. Ein Konvertierungsprogramm bildet die Brücke zu den Binärfiles. Abweichend von dem Datenschema der Datenbank, in dem eine Informationseinheit nur an einer Stelle abgelegt wird (Prinzip der Vermeidung von Datenredundanzen), ist die Datenstruktur der Binärfiles auf die häufigsten Anfragen hin optimiert. Dies führt dazu, daß in den Binärfiles Informationen mehrfach abgelegt werden und bei Modifikationen der Textfiles auf die Konsistenz der Daten geachtet werden muß.

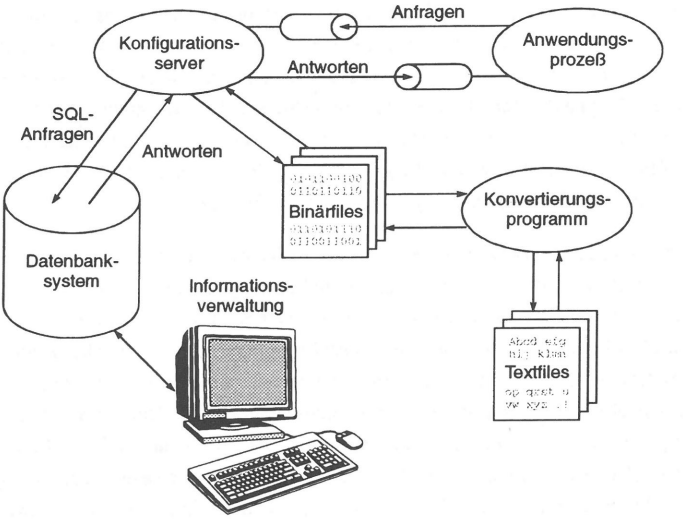


Bild 4.11: Architektur des Konfigurationsservers

4.2.4 Koordination des Zugriffes auf Meßkomponenten

Beim Einsatz des Monitoring-Systems ist es wahrscheinlich, daß verschiedene Messungen, unabhängig voneinander, periodisch Basismeßaufträge für eine Meßkomponente anstoßen. Probleme treten bei den Meßkomponenten auf, die zu einer Zeit nur einen Auftrag ausführen können [108], [123]. Trotzdem ist es notwendig, daß mehrere gleichzeitige Messungen mit diesen Komponenten durchführbar sind. Kollidierende Basismeßaufträge sind dabei zu sequentialisieren, ohne jedoch eine maximal zulässige Verzögerung des Basismeßauftrags zu überschreiten. Damit neue Meßaufträge nicht bereits bestehende Messungen in ihrer Ausführung beeinträchtigen, werden bei der Auftragserzeugung die benötigten Meßkomponenten für die gesamte Dauer einer Messung reserviert. Die Reservierungsanforderung erfolgt unter der Angabe folgender Parameter:

- einer Kennung der zugehörigen Messung (z.B. Index in die Jobliste),
- dem Startzeitpunkt der Reservierung bzw. Messung (t_{Start}),
- dem Endezeitpunkt der Reservierung bzw. Messung (t_{End}),
- einer Liste der zu reservierenden Meßkomponenten,
- der maximalen Dauer eines Basismeßauftrags (t_{BMA}) und
- der maximal zulässigen Auftragsverzögerung (t_{Del}).

Die nachfolgend beschriebene Annahmestrategie stellt sicher, daß es durch das Akzeptieren einer neuen Reservierung zu keiner Überbelegung der betreffenden Meßkomponente kommt. Dabei wird für den schlimmsten Fall davon ausgegangen, daß, wenn alle aktiven Messungen zum gleichen Zeitpunkt einen Basismeßauftrag anstoßen, die sequentielle Abarbeitung in beliebiger Reihenfolge bei keiner Messung zu einer Überschreitung der maximal zulässigen Verzögerung führen darf. Bei einer auf dieser Forderung basierenden Annahmestrategie erübrigt sich eine intelligente Einplanung der Basismeßaufträge für eine Meßstation zur Laufzeit.

Die Reservierung der Meßkomponente erfolgt beim Konfigurationsserver, der als zentrale Instanz den Überblick besitzt. Eine Reservierungsanforderung, die mehrere Meßstationen betrifft, führt entweder zu einer erfolgreichen Reservierung aller Komponenten, oder die Messung wird abgelehnt. Für die Entscheidung, ob eine Reservierung angenommen werden kann, ist die zukünftige Belegung der Meßstation aufgrund aller bereits erfolgten Reservierungen zu berücksichtigen, vor allem deren maximale Verzögerungszeiten $t_{Del}(i)$. Um für jeden Zeitpunkt die gleichzeitig aktiven Messungen bestimmen zu können, ist ein Kalender mit den Start- und Endzeitpunkten aller Reservierungen zu erstellen. Ein Kalendereintrag besteht aus einem Zeitstempel, einer Referenz auf die zugehörige Messung, dem Ereignistyp (Start bzw. Ende der Messung) und einem Belegungsgrad (t_{Load}), welcher der Gesamtdauer von jeweils einem Basismeßauftrag aller zu diesem Zeitpunkt aktiven Messungen entspricht.

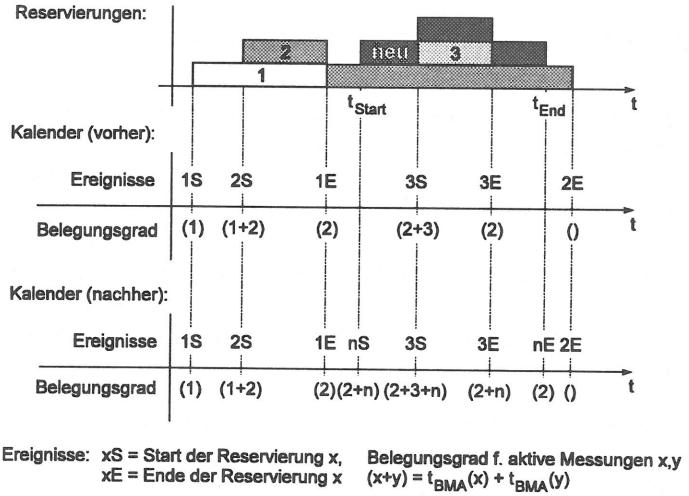


Bild 4.12: Hinzufügen einer Reservierung für eine Meßkomponente

Bild 4.12 zeigt anhand eines beispielhaften Belegungsgebirges einer Meßstation und der Kalenderzustände vor und nach einer neuen Reservierung, wie der zeitliche Verlauf des Bele-

gungsgrades ($t_{Load}(t)$) im Kalender ständig aktuell gehalten werden kann. Für das Hinzufügen einer neuen Reservierung sind in den Kalender der Beginn und das Ende der Messung einzutragen und der Belegungsgrad für alle Kalendereinträge dazwischen anzupassen, indem deren Werte um t_{BMA} erhöht werden. Beim vorzeitigen Löschen einer Reservierung können die Belegungsdauern entsprechend reduziert werden. Damit steht die Funktion $t_{Load}(t)$ zu jeder Zeit im Kalender zur Verfügung.

Für die Entscheidung, daß eine neue Reservierung angenommen werden kann, muß ($t_{Load}(t) + t_{BMA}$) für alle Zeitpunkte, die innerhalb der Reservierungsdauer liegen, kleiner als das Minimum der zulässigen Auftragsverzögerung für alle zu diesem Zeitpunkt aktiven Messungen ($t_{Del}(i)$) sein. Es muß also gelten:

$$t_{Load}(t) + t_{BMA} \leq MIN(t_{Del}(i)), \quad t_{Start} \leq t \leq t_{End}$$
$$\text{für alle Reservierungen } i, \text{ für die gilt: } t_{Start}(i) \leq t \leq t_{End}(i). \tag{4.2}$$

Die exakte Berechnung dieses Minimums ist jedoch wesentlich komplizierter als die Bestimmung von $t_{Load}(t)$. Jeder Kalendereintrag müßte hierfür eine Liste der zu diesem Zeitpunkt gleichzeitig aktiven Messungen enthalten, wobei beim Hinzufügen oder Löschen einer Reservierung diese Liste für die betreffenden Kalendereinträge zu aktualisieren wäre. Um den Berechnungsaufwand jedoch in Grenzen zu halten, wird eine restriktivere Bedingung für die Akzeptanz einer neuen Reservierung gewählt, welche alle Messungen, die während der gesamten Dauer der neuen Messung relevant sind, berücksichtigt. Sie lautet:

$$MAX(t_{Load}(t)) + t_{BMA} \leq MIN(t_{Del}(i)),$$
$$\text{wobei gilt: } t_{Start} \leq t \leq t_{End} \text{ sowie } t_{Start}(i) \leq t_{End} \wedge t_{End}(i) \geq t_{Start}. \tag{4.3}$$

Die hierfür notwendigen Werte können während eines einzigen Durchlaufs durch den Kalender bestimmt werden.

Die Zuteilung einer Meßkomponente zu einem Meßprozeß erfolgt ebenfalls zentral durch den Konfigurationsserver. Vor dem Verschicken eines Basismeßauftrags muß die Meßdurchführung ihren Belegungswunsch unter Angabe der eigenen Prozeßkennung und der voraussichtlichen Belegungsdauer anmelden. Ist der Belegungsversuch erfolgreich, muß die Meßkomponente nach der Auftragsausführung wieder freigegeben werden. Ist sie aber belegt, wird der Meßdurchführung das Ende der Belegung mitgeteilt, an dem sie ihren Belegungsversuch wiederholen muß. Dadurch wird eine Warteschlange im Konfigurationsserver für offene Belegungswünsche vermieden und der Meßprozeß kann selbst entscheiden, ob sich der Basismeßauftrag für eine Fortführung der Messung zu lange verzögert. Dies tritt jedoch aufgrund der vorherigen Reservierung nur im Fehlerfall auf.

Ist es notwendig, eine Meßstation zurückzusetzen, z.B. weil ein Basismeßauftrag nicht normal terminiert, sollte dies über den Konfigurationsserver erfolgen, damit der Rücksetzvorgang den

betroffenen Meßprozessen mitgeteilt werden kann. Dies führt dann in der Regel zu einem Abbruch der Messung, bei dem jedoch auf jeden Fall dafür zu sorgen ist, daß eventuelle Reservierungen und Belegungen der Meßkomponenten zurückgenommen werden.

4.2.5 Darstellung der Meßergebnisse

Der Anspruch, daß das Monitoring-System von verschiedenen Benutzertypen anwendbar sein soll, erfordert, daß der Detaillierungsgrad der Meßergebnisse an den Benutzer angepaßt werden kann. Die Festlegung des Abstraktionsniveaus und die Art der Präsentation, z.B. in einer Grafik, einer Tabelle oder innerhalb der Netzkarte, wird durch die entsprechende Wahl des Programms für die Ergebnisaufbereitung erreicht. Die von der Benutzungsoberfläche abhängigen Darstellungsfunktionen sind im User Interface enthalten. Die Schnittstelle zwischen Ergebnisaufbereitung und User Interface basiert auf Meldungen, die über Message Queues ausgetauscht werden. Für die darzustellenden Informationen werden nur einfache Datentypen verwendet, wie Integer-Zahlen, Arrays oder Strings. Der zeitliche Ablauf der Kommunikation ist in Bild 4.13 verdeutlicht.

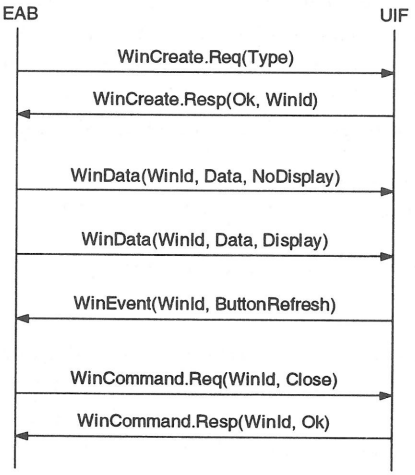


Bild 4.13: Schnittstelle zur Darstellung von Meßergebnissen

Bevor die EAB Ausgaben veranlassen kann, muß sie beim UIF ein Ausgabefenster (*Window*) anfordern (*WinCreate*). Mit dem Typ des Ausgabefensters wird der Anzeigemodus (Textliste, Tabelle, Grafik, ...) festgelegt. Unterstützt das UIF die angeforderte Fensterklasse, wird die Anforderung positiv quittiert und eine Referenz (*WinId*) übermittelt, unter dem dieses Fenster angesprochen werden kann. Daraufhin kann die EAB Daten zur Anzeige übertragen (*WinData*). Hierzu sind für jede Fensterklasse eine Reihe von Attributen definiert, aus deren

Inhalt sich die Anzeige ergibt. Für die Mitteilung von Benutzeraktionen stehen Ereignismeldungen (*WinEvent*) zur Verfügung. Besondere Aktionen, wie das Aussenden eines Alarmsignals oder die Anforderung, das Fenster zu schließen, werden von der EAB mit einer Befehlsmeldung (*WinCommand*) an das Ausgabefenster gerichtet. Wird das Fenster vom Benutzer geschlossen, erhält die EAB eine entsprechende Ereignismeldung. Dieses Verfahren paßt zu den Prinzipien der unter UNIX üblichen grafischen Oberfläche X11/Motif, wodurch die Implementierung einer grafischen Benutzungsoberfläche erleichtert wird. Es ist jedoch trotzdem so allgemein, daß auch andere Oberflächen damit realisierbar sind.

4.3 Kommunikation mit den Meßsystemen

Für den Datenaustausch mit den Meßkomponenten stützt sich die Meßdurchführung auf einen Transportdienst des Kommunikationssystems. Wird direkt ein standardisiertes Protokoll, wie TCP/IP (SNMP) oder eines der OSI-Protokolle verwendet, ist das Kommunikationssystem bereits im Betriebssystemkern enthalten. Bei RMON-basierten LAN-Monitoren z.B. beschränkt sich die Funktion der Meßdurchführung darauf, die Basismeßaufträge in entsprechende Get/Set-Aufrufe umzusetzen. Für die Abwicklung des Protokolls und die Kodierung bzw. Analyse der PDUs existieren entsprechende Funktionsbibliotheken [7], [31], [40].

Benutzt das Meßsystem jedoch ein proprietäres Meßprotokoll, sinkt die Wahrscheinlichkeit drastisch, daß auf der Plattform für das Monitoring-System eine passende Protokollimplementierung mit einer entsprechenden Programmierschnittstelle (Application Programming Interface, API) zur Verfügung steht. In diesem Fall muß das Protokoll innerhalb des Monitoring-Systems implementiert werden. Dafür ist es jedoch erforderlich, daß das Betriebssystem Anwendungsprogrammen einen Netzzugang auf der Schicht 2 zur Verfügung stellt.

Unter UNIX existieren neben dem bereits erwähnten Socket-Mechanismus weitere, universelle Mechanismen zur systemübergreifenden Interprozeßkommunikation, die weitgehend von dem Transportprotokoll abstrahieren [42], [44], [121], [122]. Andere Kommunikationsprotokolle als die aus der TCP/IP-Familie sind auf Workstations jedoch sehr selten. Einige UNIX-Derivate ermöglichen einen direkten Netzzugang auf der Schicht 2, der dem Zugriff auf normale Dateien ähnlich ist [43].

Die fehlenden Protokollfunktionen können von der Meßdurchführung selbst oder in einer speziellen Protokollinstanz zwischen Meßprozeß und Betriebssystem erbracht werden. Separate Protokollinstanzen sind insbesondere dann von Vorteil, falls Protokollfunktionen zur sicheren Informationsübertragung wie die Zeitüberwachung oder eine Wiederholung im Fehlerfall realisiert werden sollen und empfangene Pakete, wie in Bild 4.14 gezeigt, an mehrere Meßprozesse verteilt werden müssen.

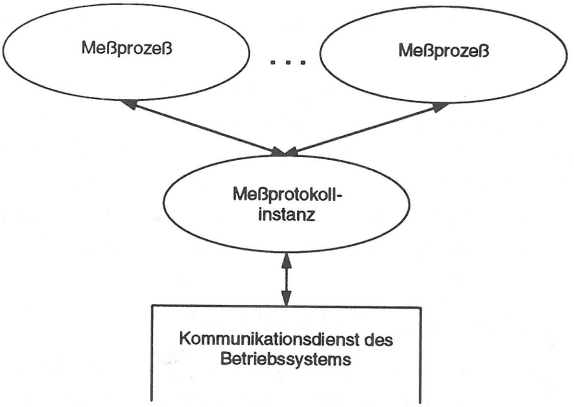


Bild 4.14: Separate Meßprotokollinstanz zur Realisierung proprietärer Meßprotokolle

Ein einfaches Meßprotokoll, basierend auf einem direkten Schicht-2-Zugang [137], wurde mit mehreren UNIX-Anwendungsprozessen realisiert [27]. Der Vorteil gegenüber einer Implementierung im Betriebssystemkern liegt in einer leichten Portierbarkeit auf andere Plattformen.

4.4 Ablauf der Messungen

Nachdem die verschiedenen Komponenten im einzelnen beschrieben sind, soll in diesem Kapitel auf das Zusammenspiel der Prozesse während der Durchführung eines Meßauftrags eingegangen werden. Der Ablauf einer Messung besteht aus fünf Phasen:

- Erzeugung der Messung,
- Start der Messung zum angegebenen Zeitpunkt,
- Ausführung der Messung,
- Anzeige bzw. Bereitstellung der Meßergebnisse und
- Beendigung sowie Entfernung der Messung aus dem System.

Bild 4.15 zeigt die möglichen Zustände einer Messung und die Prozesse, die einen Zustandswechsel herbeiführen. Der aktuelle Zustand ist im Joblistenelement abgelegt und kann jederzeit angezeigt werden. Eine nicht vorhandene Messung bzw. ein leeres Element in der Jobliste besitzt den Zustand free. Zu einem solchen Element existiert auch kein Meßjob. Dieser wird erst angelegt, nachdem die Auftragserzeugung (AEZ) ein Joblistenelement belegt, indem sie dessen Zustand nach in_creation ändert. Nachdem der Meßjob vollständig ausgefüllt ist, setzt sie die Messung in den Zustand ready. Messungen, die augenblicklich starten sollen, setzt der Monitoring Scheduler direkt auf running, während zu verzögernde Messungen über den Zustand waiting dorthin gelangen. Am Ende der Messung setzt die Meßdurchführung den Zustand auf

finished, in welchem die Messung so lange verbleibt, wie die Meßergebnisse benötigt werden. Die Messung wird schließlich vollständig aus dem System entfernt, indem die Ergebnisaufbereitung (EAB) den Zustand wieder auf free setzt und den Meßjob löscht.

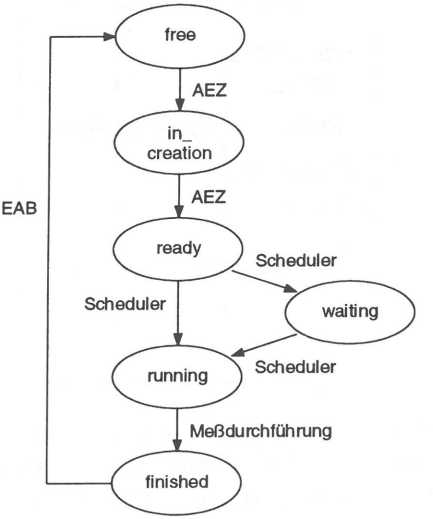


Bild 4.15: Zustandsdiagramm einer Messung

Die Zugriffsrechte auf den Meßjob sind über den Zustand der Messung geregelt, ohne daß dies durch besondere Maßnamen sichergestellt wird. Während des Zustandes in_creation besitzt die AEZ ein exklusives Zugriffsrecht. Befindet sich die Messung in der Ausführung (running), darf die Meßdurchführung die Parameter auslesen und die Meßergebnisse im Meßjob ablegen, die wiederum von der EAB ausgelesen werden dürfen. Schreib/Lesekonflikte können dabei nur auftreten, wenn die Meßdurchführung verschiedene Ergebnisse in denselben Ergebnisbereich schreibt (Modus Overwrite). Deshalb legt die Meßdurchführung zum Ergebnis die laufende Nummer der Messung mit ab. Die EAB kann erkennen, daß sie konsistente Ergebnisse gelesen hat, wenn die Ergebnisnummer vor und nach dem Auslesen denselben Wert besitzt.

Den Ablauf der Erzeugung einer Messung für den Fall, daß keine Fehler auftreten, ist in Bild 4.16 dargestellt. Er beginnt damit, daß der Benutzer einen Meßauftrag generiert und ihn der AEZ übergibt. In Zusammenarbeit mit dem Konfigurationsserver werden die Stationsadressen bestimmt. Danach belegt die AEZ ein freies Element in der Jobliste. Ist der Meßauftrag vollständig erzeugt und sind die beteiligten Meßkomponenten reserviert, wird die Messung in den Zustand ready versetzt und dies dem Monitoring Scheduler über ein Semaphor mitgeteilt.

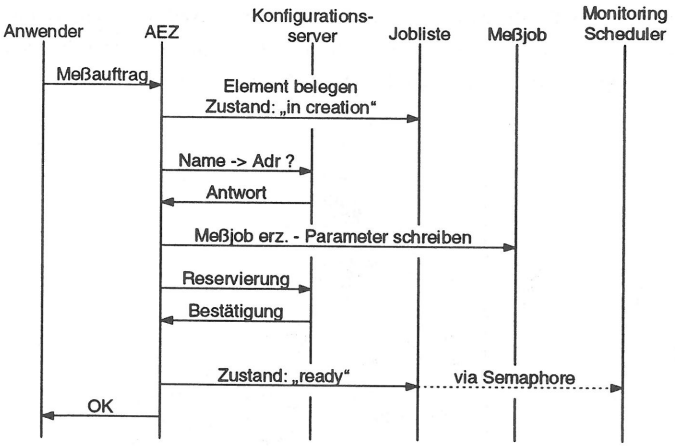


Bild 4.16: Erzeugung einer Messung

Der Monitoring Scheduler befindet sich die meiste Zeit wartend an dem Semaphor für neue Meßaufträge. Nach dem Verlassen dieses Zustandes muß der Scheduler zuerst die betroffene Messung in der Jobliste suchen. Liegt ein späterer Startzeitpunkt der Messung vor, wird ein Timer gesetzt und an dem Semaphor wieder auf neue Messungen gewartet. Tritt ein Timeout auf oder ist die Messung sofort zu starten, wird der entsprechende Meßprozeß erzeugt und ggf. ein Prozeß für die EAB gestartet. Den Ablauf zeigt Bild 4.17.

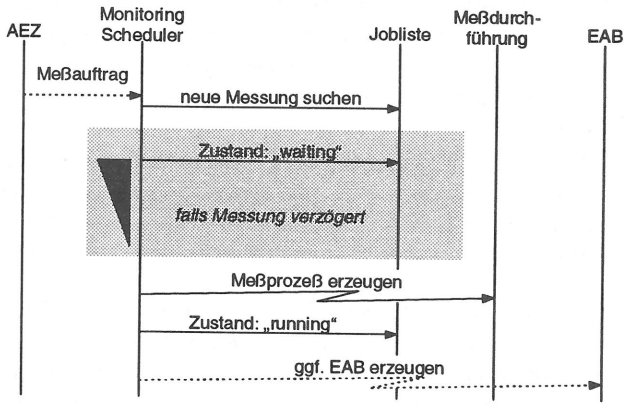


Bild 4.17: Verzögerung und Start einer Messung

Die Meßdurchführung bekommt die Nummer des Joblistenelements als Aufrufparameter vom Monitoring Scheduler mitgeteilt, mit deren Hilfe sie über die Jobliste die Systemkennung des Meßjobs ermittelt. Es folgen das Auslesen und Auswerten der Meßauftragsparameter und die Anmeldung beim Kommunikationssystem. Bild 4.18 zeigt den Ablauf einer periodischen Messung, bestehend aus einem Basismeßauftrag. Abhängig von dem Startzeitpunkt der EAB und der Behandlung der Meßergebnisse werden diese im Meßjob abgespeichert oder an die Message Queue der EAB geschickt. Am Ende der Messung wird die Verbindung zum Kommunikationssystem wieder abgebaut, der EAB das Ende der Messung angezeigt bzw. die EAB erzeugt, das Ende der Messung in die Jobliste eingetragen und die Reservierung der Meßstationen, die von der AEZ durchgeführt wurde, wieder aufgehoben.

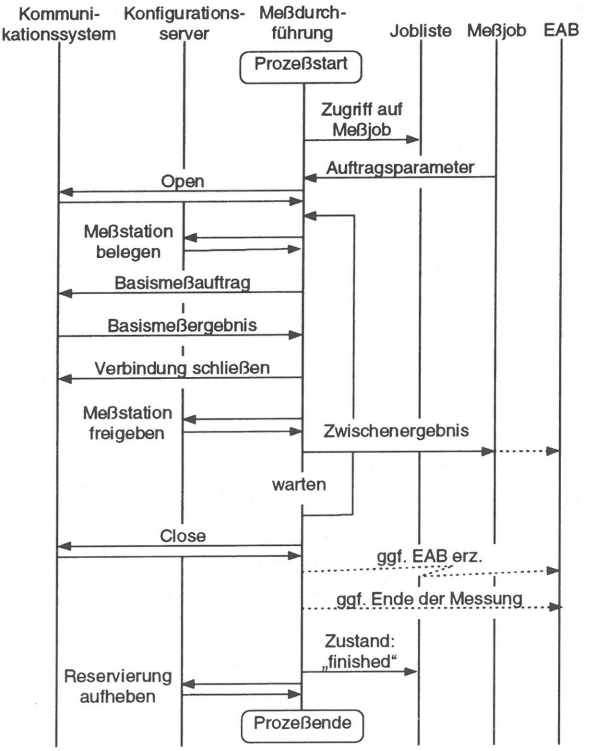


Bild 4.18: Ausführung der Messung

Die Aufbereitung der Ergebnisse einer Messung hängt ebenso wie deren Ausführung von der Art der Messung ab. So können wahlweise Zwischenergebnisse über die Message Queue eintreffen, oder sie werden periodisch aus dem Meßjob ausgelesen. Darüber hinaus kann der Be-

nutzer über das UIF die Ausgabe eines aktuellen Ergebnisses anstoßen. Bild 4.19 zeigt diese Möglichkeiten beispielhaft in einem Diagramm.

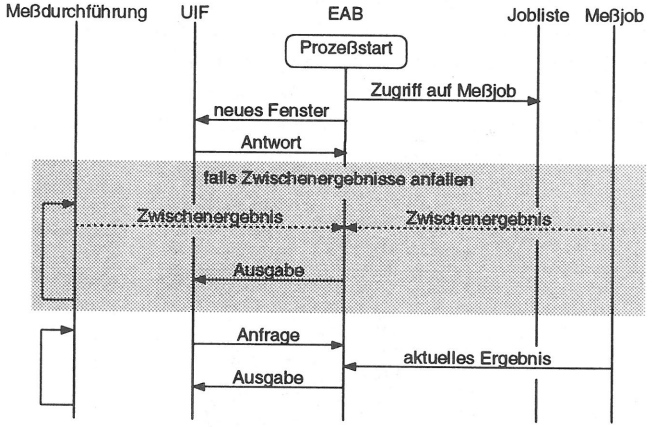


Bild 4.19: Ergebnisaufbereitung und -ausgabe

Das Ende der Messung ist in Bild 4.20 dargestellt. Existiert die EAB bereits während der Messung, wird sie von der Meßdurchführung über das Ende informiert und stellt das Endergebnis dar. Erst wenn die Ergebnisse nicht mehr benötigt werden, d.h. der Benutzer das Ausgabefenster schließt, wird der Joblistenelement freigegeben und der Meßjob gelöscht.

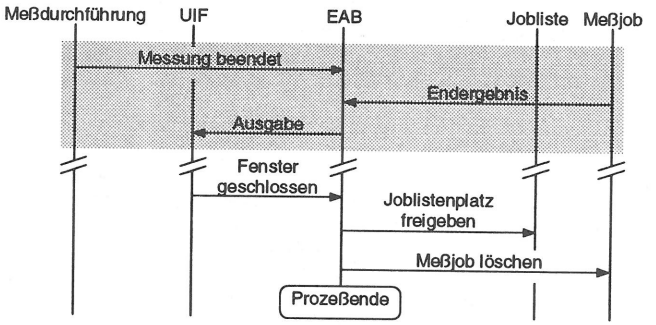


Bild 4.20: Entfernung der Messung aus dem System

4.5 Realisierung durch einen Prototyp

Mit der Entwicklung eines Prototyps für das Monitoring-System konnte das in Kapitel 3 vorgestellte Überwachungskonzept seine Tragfähigkeit beweisen, und es wurden die in Kapitel 4.2 beschriebenen Konzepte auf ihre Implementierbarkeit hin überprüft.

4.5.1 Allgemeine Randbedingungen

Die Entwicklung des Monitoring-Systems entsprang aus einer Kooperation des Instituts für Nachrichtenvermittlung und Datenverarbeitung und der Firma Siemens AG, Bereich Automatisierungstechnik. Primäres Ziel war die zentrale Verwaltung und Steuerung eines in einem Vorläuferprojekt vom Institut entwickelten verteilten Meßsystems [125], [128]. Diese Kooperation lieferte die Motivation für die umfassende Auseinandersetzung mit der Überwachungsproblematik von Rechnernetzen, und führte zu einem wesentlich universeller einsetzbaren System, als ursprünglich angedacht. Bei dem verteilten Meßsystem handelt es sich um modular aufgebaute Meßstationen, bestehend aus einer Prozessorkarte mit Netzanschaltung (Basiskarte), die über einen internen Systembus verschiedene Meßmodule steuert. Zu den Meßmodulen gehören u.a. eine Reflektometerbaugruppe zur Störstellenerkennung an Ethernet-Segmenten während des Netzbetriebs [8] und eine Durchflußmesserbaugruppe, welche, eingeschleift in die Verbindung der Station zum Netzzugangspunkt (Transceiver Access Point, TAP), Netzaktivitäten und Kollisionen getrennt nach Sende- und Empfangsrichtung erfassen kann. Bild 4.21 zeigt das Blockschaltbild einer Meßstation.

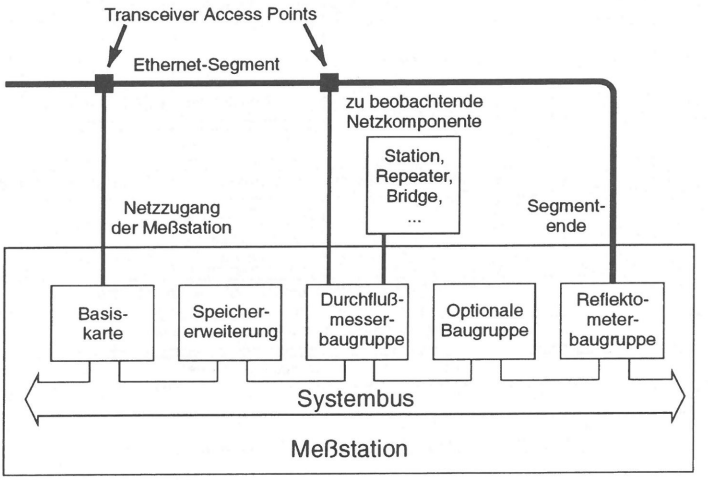


Bild 4.21: Blockschaltbild einer Meßstation

Die Implementierung des Monitoring-Systems erfolgte auf einem Personal Computer (PC) unter dem Betriebssystem Open Desktop 1.1 (ODT), einem UNIX-Derivat der Firma SCO in der Programmiersprache C. Für die Konfigurationsdatenbank kam das in ODT enthaltene Datenbanksystem INGRES zum Einsatz.

4.5.2 Struktur der Software

Das Monitoring-System besteht aus einer Reihe von Programmen, die von den verschiedenen Prozessen ausgeführt werden. Während der Initialisierung des Systems werden die zentralen Prozesse gestartet und stehen ständig im Hintergrund zur Verfügung. Die zentralen Prozesse sind neben dem Kommunikationssystem [27], der in Kapitel 4.5.4 beschriebene zentrale Message Handler, der Monitoring Scheduler [100] und der Konfigurationsserver [21]. Das User Interface wird nicht ständig benötigt. Je nach Anforderung des Benutzers kann eine grafische oder zeichenorientierte Oberfläche gestartet werden. Es ist möglich, neue Meßaufträge durch direkten Aufruf der Auftragserzeugung zu generieren. Die übrigen Prozesse werden im Rahmen des Ablaufs der Messungen gestartet und kontrolliert. Jedem Prozeß liegt ein eigenständiges Programm zugrunde.

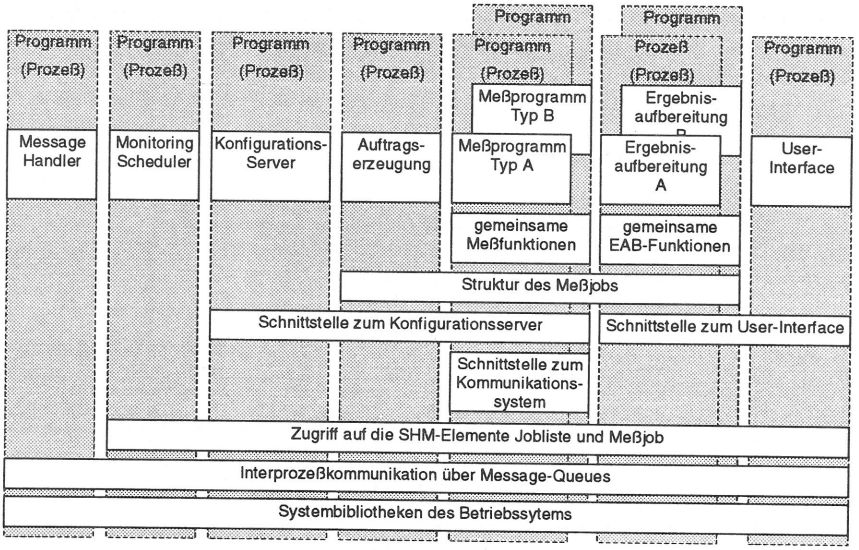


Bild 4.22: Struktur der Software des Monitoring-Systems

Die Struktur dieser Programme besteht, wie in Bild 4.22 dargestellt, aus Modulen, die eine Schichtung aufweisen. Die Namen der Module der obersten Ebene entsprechen den Prozessen.

Mehrfach benötigte Funktionen sind in gemeinsam benützte Module ausgelagert. Da die Implementierung nicht objektorientiert erfolgt, werden allgemeine Mechanismen wie der korrekte Zugriff auf die SHM-Elemente oder die Interprozeßkommunikation über Message Queues in Module gekapselt und über eine Funktionsschnittstelle den darüberliegenden Schichten angeboten. Die Einhaltung der Schnittstellen zwischen den Prozessen wird ebenfalls durch eine Kapselung in Funktionen, die von beiden Instanzen verwendet werden, gesichert.

Die Vielfalt der Messungen schlägt sich in der Anzahl der unterschiedlichen Programme für die Meßdurchführung und die Ergebnisaufbereitung nieder. Die Auslagerung gemeinsamer Funktionalitäten in zentrale Module erlaubt, sich bei der Entwicklung neuer Programme auf das Wesentliche zu konzentrieren und verringert den Implementierungsaufwand beträchtlich. Globale Konzepte für die Ausgabe von Meldungen und für die Fehlerbehandlung vereinfachen nicht nur die Entwicklungsphase, sondern sind entscheidend für die spätere Wartbarkeit des Softwaresystems.

4.5.3 Kommunikation zwischen den Prozessen

Für die Kommunikation zwischen den Prozessen, die einem Meßauftrag zugeordnet sind, können alle erforderlichen Informationen über die globale Jobliste ermittelt werden. Die Zugriffsfunktionen auf die Jobliste stellen sicher, daß das Zugriffsprotokoll, d.h. die korrekte Verwendung des Joblisten-Semaphors, eingehalten wird. Zusätzlich ist ein Informationsaustausch über Meldungen erforderlich. Dieser wird mit Hilfe der Inter-Module-Communication-Schnittstelle (IMC) abgewickelt, die von dem darunterliegenden IPC-Mechanismus, hier den Message Queues, abstrahiert. Die Funktionalität der IMC-Schnittstelle besteht im Transfer von Meldungen von einem Modul (Prozeß) zum einem anderen. Für die Adressierung besitzt jeder Prozeß eine eindeutige *Modul-Id*. Jede IMC-Meldung, deren Struktur Bild 4.23 zeigt, besitzt einen Kopf, der die Modul-Ids des Senders und Empfängers enthält. Die *Primitive-Id* legt das Format der nachfolgenden Daten fest.

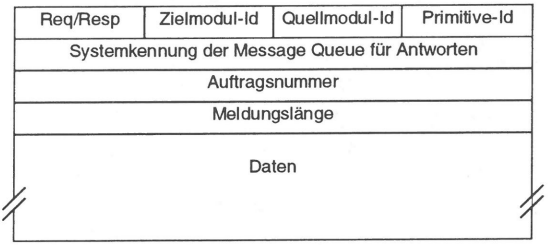


Bild 4.23: Der IMC-Meldungskopf

Das Konzept mit eindeutigen Modul-Ids eignet sich für eine Client-Server-Kommunikationsbeziehung zwischen systemweit nur einmal existierenden Prozessen. Damit auch mehrfach existierende Client-Prozesse mit einem zentralen Server kommunizieren können, wurde als Modul-Id für mehrfach existierende Prozesse ein reservierter Wert gewählt. Bei Request-Meldungen von diesen Prozessen entnimmt der Server die Systemkennung der Message Queue für die Antwort direkt aus dem Auftrag. Beim Meldungsaustausch zwischen zwei mehrfach existierenden Prozessen, wie z.B. die Meßdurchführung und die Ergebnisaufbereitung, muß der sendende Prozeß die Systemkennung der Zielwarteschlange über andere Mechanismen (hier die Jobliste) ermitteln. In diesem Fall wird jedoch über die Schnittstelle der darunterliegende IPC-Mechanismus nicht mehr vollständig verborgen.

Der Ablauf der Kommunikation zwischen zwei Prozessen über die IMC-Schnittstelle ist in Bild 4.24 dargestellt. Die Funktion *imc_init()* muß vom jedem Prozeß einmal aufgerufen werden und legt die Empfangswarteschlange an, wobei die Modul-Id in den Key-Value der Message Queue mit einfließt. Mit *imc_bind()* wird die Systemkennung der Message Queue des Partnerprozesses ermittelt. Danach kann mit *imc_send()* die Meldung verschickt werden. Beim Aufruf der Funktion *imc_receive()* wartet der Prozeß so lange, bis eine Meldung in seiner Empfangswarteschlange eintrifft, wobei eine maximale Wartezeit angegeben werden kann.

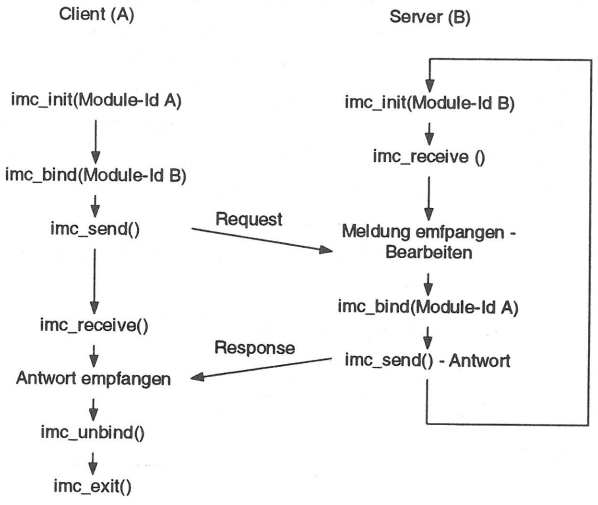


Bild 4.24: Kommunikationsablauf über die IMC-Schnittstelle

4.5.4 Verwaltung von System- und Fehlermeldungen

Durch die Aufteilung des Monitoring-Systems in mehrere Prozesse sind die standardmäßigen Kanäle für die Ausgabe von Meldungen an den Benutzer nicht mehr verwendbar. Über einen zentralen *Message Handler*, der die Meldungen aller Prozesse in einer Message Queue entgegennimmt, ist es möglich, einheitlich auf diese Meldungen zu reagieren und z.B. an das User Interface zur Anzeige zu verschicken. Läuft das System unbeaufsichtigt, müssen die Meldungen anstatt angezeigt in einer Datei protokolliert werden.

Zur selektiven Bearbeitung werden die Meldungen in verschiedene Klassen eingeteilt. Ein Prozeß, der bestimmte Meldungen in seiner Empfangswarteschlange erhalten möchte, meldet sich beim Message Handler für die ihn interessierenden Meldungsklassen an. Alternativ kann ein Dateiname angegeben werden, auf den die Meldungen protokolliert werden sollen. Falls erforderlich werden die Meldungen vervielfacht und an mehrere Interessenten verteilt.

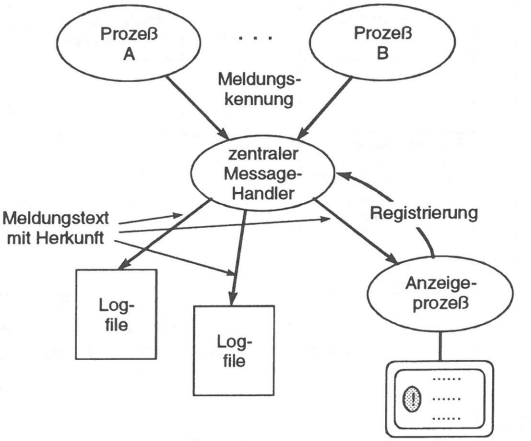


Bild 4.25: Architektur des zentralen Message Handlers

Bild 4.25 zeigt die Architektur des zentralen Message Handlers. Um die Meldungstexte von den Prozessen zu trennen, werden lediglich Meldungskennungen an den Message Handler übermittelt, der den Meldungstext, Schweregrad und den Namen des Absenderprozesses aus einer zur Laufzeit eingelesenen Datei bestimmt, formatiert und entsprechend weiterleitet.

4.5.5 Meldungsschnittstelle zum Konfigurationsserver

Die Meldungsschnittstelle zwischen der Auftragserzeugung bzw. der Meßdurchführung und dem Konfigurationsserver stellt sowohl vorgefertigte Abfragen zur Verfügung, die unter Ver-

wendung der Informationen aus den Binärfiles sehr effizient bearbeitet werden können, ermöglicht aber auch allgemeine Abfragen der Konfigurationsdatenbank basierend auf SQL, die als Zeichenkette übertragen werden. Eine vorgefertigte Abfrage legt mit dem Abfragetyp die Entitätsmenge und die zulässigen Kombinationen der Suchfelder für die Selektion der auszuwählenden Datensätze fest. Die Meldungen enthalten, wie in Bild 4.26 gezeigt, eine Suchfeldliste und eine Datenliste. Die Suchfeldliste enthält Bedingungen, die ein Datensatz erfüllen muß, um ausgewählt zu werden, während die Elemente der Datenfeldliste die Attribute bezeichnen, die in der Antwort enthalten sein sollen. Besitzt in einer Auftragsmeldung ein Element der Datenfeldliste einen Wert, wird dieser in die Konfigurationsdatenbank geschrieben.

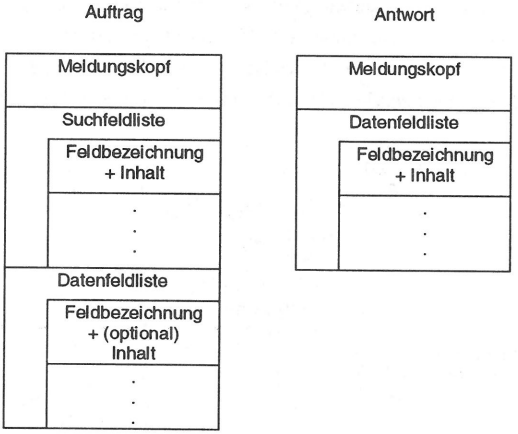


Bild 4.26: Meldungsschnittstelle zum Konfigurationsserver für vorgefertigte Abfragen

4.5.6 Benutzungsoberfläche

Für den Prototyp des Monitoring-Systems wurde eine zeichenorientierte und eine graphische Benutzungsoberfläche realisiert. Bedingt durch den nebenläufigen Charakter der Netzüberwachung sind auf dem Bildschirm unterschiedliche Dinge mehr oder weniger gleichzeitig darzustellen. Deshalb ist die zeichenorientierte Oberfläche mit Hilfe eines semigraphischen Fenstersystems [98] realisiert, welches zu diesem Zwecke den Bildschirm in verschiedene Bereiche einteilt, in denen die Meldungen des zentralen Message Handlers, die Reaktionen auf Benutzereingaben und Informationen über eintreffende Zwischenergebnisse unabhängig voneinander ausgegeben werden. Bei Bedarf kann auch eine zyklische Anzeige des Zustandes der im System befindlichen Messungen in einem Fenster erfolgen. Meßergebnisse, die über die Schnittstelle in Kapitel 4.2.5 beschriebene Schnittstelle zwischen Ergebnisaufbereitung und User Interface eintreffen, werden in getrennten Fenstern angezeigt, die über das Schnittstellenprokotoll angelegt und gelöscht werden. Zur Gesamtüberwachung eines größeren Netzes ist

jedoch eine graphische Benutzungsoberfläche unumgänglich. Basierend auf der Programmierschnittstelle von Motif wurde eine graphische Benutzungsoberfläche für das Monitoring-System entwickelt [129]. Bild 4.27 soll einen Eindruck vermitteln, wie sich die Bedienung des Monitoring-System gestaltet.

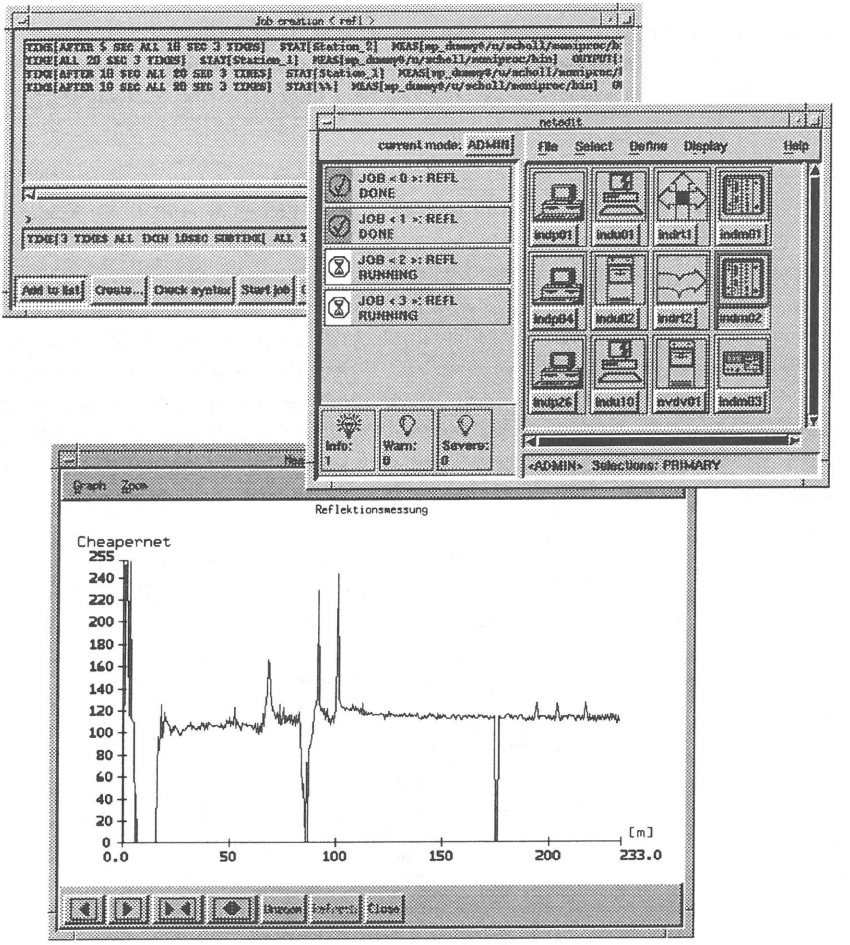


Bild 4.27: Grafische Benutzungsoberfläche des Monitoring-Systems

In dem Hauptfenster in der Mitte des Bildes sind links die im System befindlichen Messungen dargestellt und rechts die Meßstationen und die übrigen Netzkomponenten. Die drei Felder

links unten informieren über neu eingetroffene Nachrichten des zentralen Message-Handlers. Nach der Auswahl einer Meßstation kann im oberen Fenster aus einer Liste von vorgefertigten Meßaufträgen einer ausgewählt und gestartet werden. Der neue Meßauftrag erscheint dann links im Hauptfenster, wo er jederzeit selektiert werden kann, um sich die Meßparameter und Meßergebnisse in Textform darstellen zu lassen oder die Messung abzubrechen. Das untere Fenster zeigt das Ausgabefenster der Ergebnisaufbereitung einer Reflektionsmessung, in der die Reflektionskurve eines Ethernet-Segmentes dargestellt ist. Darüber hinaus ist es möglich, stationsindividuelle Daten wie z.B. die Erreichbarkeit direkt über die Stationsobjekte im Hauptfenster anzuzeigen.

Schließlich ist noch die direkte Benutzung des Monitoring-Systems aus der Kommandozeilenebene des Betriebssystems heraus (Shell) zu erwähnen. Die Bedienung erfolgt über Kommandos mit denen Programme aufgerufen werden, die lediglich über die Standard-Ein/Ausgabekanäle mit dem Anwender kommunizieren. Ein solches Programm ist die Auftragserzeugung, andere erlauben die Anzeige des Systemzustandes und der Jobliste bzw. der Meßjobs. Mit speziellen Programmen für die Ergebnisaufbereitung werden die Meßergebnisse kurz und lesbar auf das Terminal ausgegeben bzw. in einen File geschrieben. Dabei erscheinen die Ergebnisse entweder spontan auf dem Ausgabekanal oder sie werden explizit durch den Start der EAB abgefragt. Diese Art der Bedienung ist sicherlich nicht geeignet, viele periodische Langzeitmessungen parallel durchzuführen, stellt jedoch eine einfache und effiziente Methode dar, aus einer Applikation heraus, Messungen mit dem Monitoring-System durchzuführen.

Kapitel 5

Leistungsuntersuchung des Monitoring-Systems

Grundlage für die Akzeptanz eines Softwaresystems beim Anwender ist, daß die Programmlaufzeiten auf der Zielplattform gewisse Grenzen nicht überschreiten. Bei interaktiven Programmabläufen liegen die dem Benutzer zumutbaren Reaktionszeiten im Bereich von einigen Zehntelsekunden bis wenigen Sekunden und zwar unabhängig davon, wie stark der Prozessor durch Hintergrundaktivitäten ausgelastet ist. Überwachungssysteme wie das Monitoring-System zeichnen sich dadurch aus, daß bei den einzelnen Überwachungsvorgängen zeitliche Randbedingungen eingehalten werden müssen. Diese Bedingungen gelten jedoch nicht so streng wie in Echtzeitsystemen, da in Hochlastsituationen durchaus selten stattfindende Überschreitungen toleriert werden, d.h. keine wirklich harten Zeitbedingungen existieren. Wie bereits in Kapitel 4.1.5 diskutiert, garantieren zwar Echtzeitbetriebssysteme beschränkte Verzögerungszeiten auf externe Ereignisse und beschränkte Bearbeitungsdauern zeitkritischer Prozesse, es kann jedoch effektiver sein, ein nicht echtzeitfähiges Betriebssystem zu verwenden, wenn damit der Einsatz leistungsfähigerer Rechnerplattformen ermöglicht wird.

Die Leistungsuntersuchung in diesem Kapitel soll klären, bis zu welchem Grad das Monitoring-System mit Messungen beaufschlagt werden darf, ohne daß es zu nicht mehr tolerierbaren Verzögerungen in der Meßausführung kommt. Zusätzlich sollen dabei mögliche Engpässe im System erkannt werden. Die Möglichkeit, diese Untersuchungen durch Messungen an einem Prototypen durchzuführen, scheidet aus zwei Gründen aus. Zum einen können realistische Lastsituationen im Labor nur sehr schwer erzeugt werden, da hierfür ein großes Testnetz mit vielen Meßkomponenten erforderlich ist und die prototypische Realisierung vieler Meßfunktionen einen hohen Implementierungsaufwand bedeutet. Zum anderen können die Prozeßabläufe in einem realen System nur sehr schwer beobachtet werden, ohne das System selbst zu beeinflussen. Deshalb wurde das Monitoring-System simulativ untersucht, wobei wie folgt vorgegangen wurde:

- Die Prozesse des Monitoring-Systems und deren Abläufe wurden durch ein verkehrstheoretisches Modell nachgebildet.
- Anhand gemessener Programmlaufzeiten wurden die Bediendauern der Modellkomponenten abgeschätzt.
- Ausgehend von verschiedenen Meßabläufen wurde das Monitoring-System simuliert und die interessierenden Größen gemessen.

Über die Simulation des Monitoring-Systems wird auch das korrekte Zusammenspiel der Prozesse validiert, wobei dies jedoch keine Verifikation des Konzeptes darstellt.

5.1 Modellierung des Monitoring-Systems

Üblicherweise werden Leistungsuntersuchungen bei Systemen durchgeführt, die aus mehreren, voneinander unabhängigen Bedieneinheiten bestehen, welche untereinander Meldungen austauschen. Dabei wird angenommen, daß die Bedienzeiten von den Meldungen unabhängig sind und über eine statistische Verteilungsfunktion abgeschätzt werden können. Beim Monitoring-System entsprechen jedoch diese Bedieneinheiten den Betriebssystemprozessen, die wiederum in erheblichem Maße von den Messungen abhängig sind und über das Joblisten-Semaphor und den Meldungsaustausch untereinander korreliert werden. Für die Untersuchung des zeitlichen Verhaltens der Meßaufträge ist es deshalb notwendig, die einzelnen Bearbeitungsschritte sehr genau zu modellieren, besonders die Interprozeßaktivitäten.

5.1.1 Überblick über das Gesamtmodell

Im Simulationsmodell des Monitoring-Systems werden die Meßaufträge durch Meldungen verkörpert, welche alle Meßparameter, wie die Anzahl der Teilmessungen oder den Startzeitpunkt der Ergebnisaufbereitung, enthalten. Für jede Klasse von Meßaufträgen existiert ein eigener Generator, der die Aufträge mit einer bestimmten Ankunftsrate erzeugt und variable Parameter, wie z.B. den Beginn der Messung, zufällig, basierend auf einer vorgegebenen statistischen Verteilungsfunktion, bestimmt. Während der Bearbeitung eines Meßauftrages durchlaufen die Meldungen die verschiedenen Prozesse des Monitoring-Systems entsprechend dem wirklichen Ablauf. Bild 5.1 zeigt das gesamte Simulationsmodell des Monitoring-Systems in einer Übersicht. Den dort dargestellten Rechtecken liegen Warteschlangenmodelle zugrunde. Sie bilden die einem Prozeß des Monitoring-Systems entsprechenden Teilmodelle und sind in Kapitel 5.1.3 genauer beschrieben. Pfeile deuten den Meldungsfluß an. Da für jeden dynamisch erzeugten Prozeß im Simulationsmodell eine Bedieneinheit vorgesehen werden muß, existieren die Modelle der Meßdurchführung und der Ergebnisaufbereitung paarweise für jedes Element in der Jobliste einmal. Wie im realen System ordnet die Auftragserzeugung einer neuen Messung ein freies Joblistenelement und damit ein Prozeßpaar zu. Ist kein freies Element verfügbar, wird der Meßauftrag in der Auftragserzeugung verworfen.

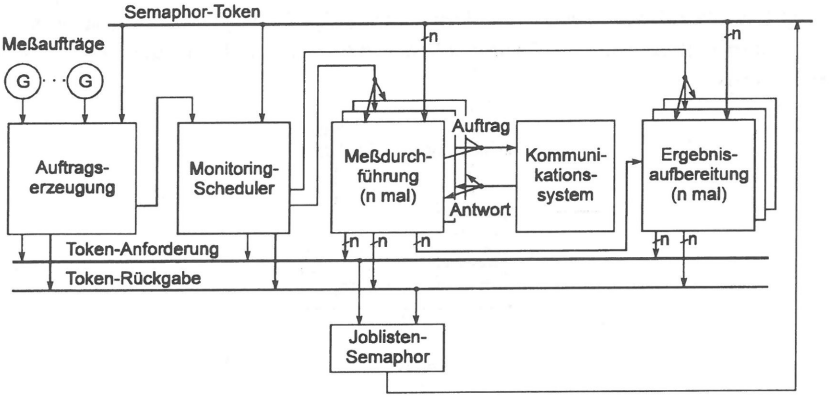


Bild 5.1: Übersicht des Simulationsmodells für das Monitoring-System

Die Synchronisation der Prozesse untereinander erfolgt über die ausgetauschten Meldungen. So wird z.B. eine Meßdurchführung erst ablauffähig, nachdem sie einen Meßauftrag vom Monitoring Scheduler erhalten hat. Die Prozeßsynchronisation über das Joblisten-Semaphor wird über einen in sich geschlossenen Meldungsfluß mit den Meldungstypen *Semaphor-Token*, *Token-Anforderung* und *Token-Rückgabe* und den Joblisten-Semaphor modelliert. Damit ein Prozeß auf die Jobliste zugreifen kann, muß er eine Token-Anforderung an den Joblisten-Semaphor schicken und auf die Zuteilung des Tokens warten. Ist das Token gerade in Besitz eines anderen Prozesses, wird die Anforderung in einer Warteschlange so lange gespeichert, bis das Joblisten-Semaphor das Token über eine Token-Rückgabemeldung wieder erhält.

Auf eine Modellierung des Konfigurationsservers wurde verzichtet, da er immer im Wechsel zu einem Prozeß, dessen Anforderung er gerade bearbeitet, abläuft. Seine Bedienzeiten können deshalb dem anfordernden Prozeß zugerechnet werden, ohne den Gesamtablauf wesentlich zu beeinflussen.

5.1.2 Allgemeine Komponenten

5.1.2.1 Joblisten-Semaphor

Das Joblisten-Semaphor wurde nicht als Bedieneinheit modelliert, da eventuelle Bediendauern in den Bedienzeiten der Prozesse enthalten sind. Das Joblisten-Semaphor synchronisiert eine Token-Anforderung mit der Token-Rückgabemeldung eines anderen Prozesses und vereinigt die beiden Meldungen zu einer Semaphor-Token-Meldung. Das Semaphor-Token kehrt schließlich wieder als Token-Rückgabemeldung zurück. Das Warteschlangenmodell des Joblisten-Semaphors ist in Bild 5.2 dargestellt. Das Symbol für die Synchronisation wurde aus der

Theorie der Petri-Netze entlehnt (vgl. hierzu auch [28]). Zu Beginn der Simulation ist die rechte Warteschlange mit einem Semaphor-Token initialisiert, welches im weiteren Verlauf im System kreist.

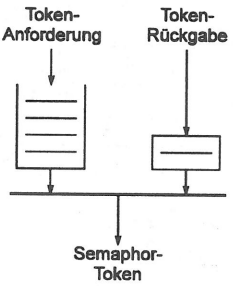


Bild 5.2: Warteschlangenmodell eines Semaphors

5.1.2.2 Variables Verzögerungsglied

Zeitlich begrenzte Ausführungsunterbrechungen eines Prozesses, wie z.B. für Verzögerung des Meßauftrages beim Monitoring Scheduler, bis der Startzeitpunkt der Messung erreicht ist, werden im Simulationsmodell durch ein variables Verzögerungsglied, wie es in Bild 5.3 dargestellt ist, erreicht.

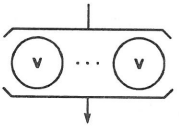


Bild 5.3: Variables Verzögerungsglied

Dieses Element speichert eintreffende Meldungen für eine gewisse Dauer, um sie danach unverändert weiterzusenden. Da beliebig viele Meldungen gleichzeitig verzögert werden können, ist das Element im Simulationsmodell als *Infinite Server* dargestellt. Das variable Verzögerungsglied entnimmt die Verzögerungsdauer aus der Meldung.

5.1.3 Prozesse des Monitoring-Systems

Basierend auf den Ablaufdiagrammen aus Kapitel 4.4 werden die Zugriffe auf den Joblisten-Semaphor und der Meldungsaustausch zwischen den Prozessen detailliert nachgebildet. Hierzu wird ein Prozeß, der im Prinzip einer Bedienphase entspricht, in mehrere Teilphasen aufgespalten, die einzelne Programmabschnitte zwischen zwei Interprozeßkommunikationsaktivitäten

nachbilden. Die Tatsache, daß in einem Prozeß zu einem Zeitpunkt nur eine Teilphase ablauffähig sein kann, wird im Warteschlangenmodell durch eine spezielle Meldung, das Ablauf-Token, sichergestellt. Diese Meldung verhindert z.B., daß wenn am Eingang der ersten Teilphase mehrere Aufträge zur Bearbeitung anstehen, diese Teilphase zur Ausführung kommt, bevor der laufende Auftrag komplett abgearbeitet ist. Die Teilphasen aller Prozesse sind als Bedienphasen einer einzigen Bedieneinheit zu verstehen. Diese Bedieneinheit entspricht dem Prozessor des Rechnersystems, auf dem das Monitoring-System abläuft.

5.1.3.1 Auftragserzeugung

Das detaillierte Warteschlangenmodell der Auftragserzeugung zeigt Bild 5.4. Die externen Ein- und Ausgänge entsprechen denen in Bild 5.1.

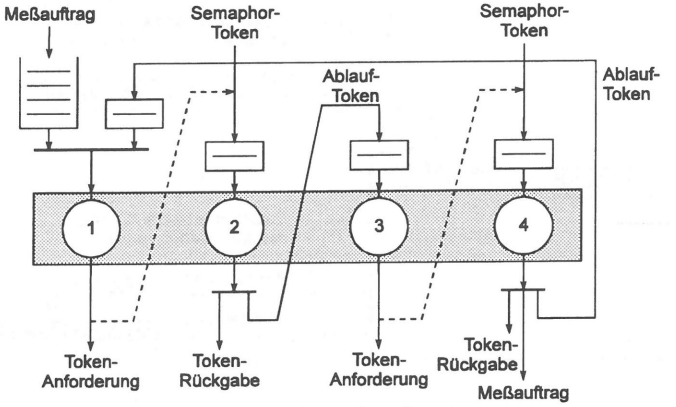


Bild 5.4: Warteschlangenmodell der Auftragserzeugung

Bevor die Phasen 2 und 4, die auf die Jobliste zugreifen, ablaufen können, ist jeweils ein Token aus dem Joblisten-Semaphor anzufordern, wodurch der Prozeß unterbrochen wird. Der Erhalt des Semaphor-Tokens ist durch einen gestrichelten Pfeil dargestellt. Die neuen Meßaufträge landen in der linken Eingangswarteschlange der Teilphase 1. Das Synchronisationselement stellt sicher, daß der nächste Meßauftrag den Prozeß erst belegen kann, nachdem der vorherige am Ende der Phase 4 an den Monitoring Scheduler weitergeschickt wurde. An den Ausgängen der Teilphasen 2 und 4 werden Meldungen, die zuvor in Synchronisationselementen vereinigt wurden, wieder vervielfacht. Den Ablauf der Teilphasen bei der Erzeugung eines Meßauftrags und die dabei ausgetauschten Meldungen beschreibt Bild 5.5.

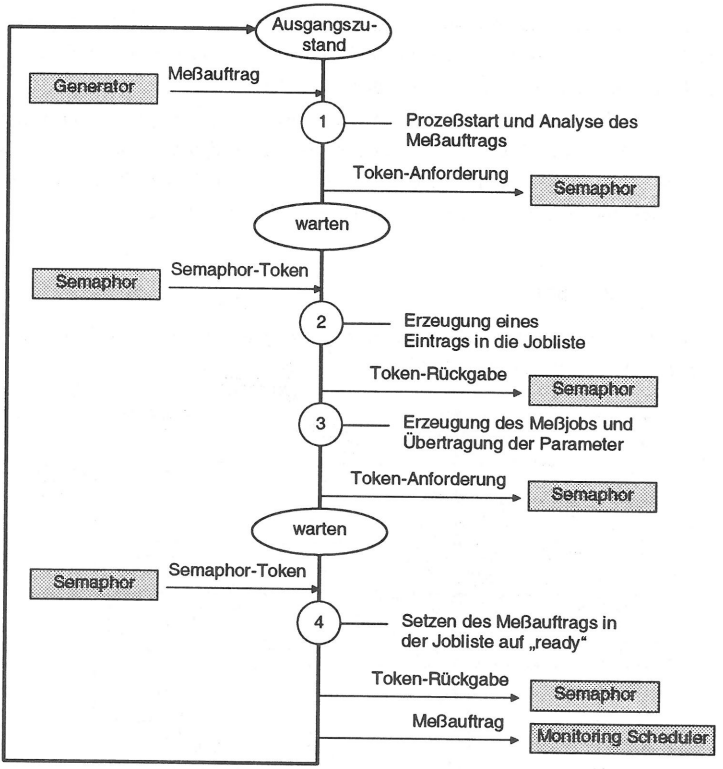


Bild 5.5 Ablauf der Teilphasen bei der Auftragserzeugung

5.1.3.2 Monitoring Scheduler

Das Warteschlangenmodell des Monitoring Schedulers ist in Bild 5.6 dargestellt. Es besteht aus vier Teilphasen und einem variablen Verzögerungsglied. In der ersten Teilphase wird erkannt, daß ein neuer Meßauftrag eingegangen ist. In Teilphase 2 erfolgt der Zugriff auf die Jobliste, und es wird abhängig davon, ob der Meßauftrag verzögert werden muß oder nicht, die Meßauftragsmeldung an die entsprechende Meßdurchführung weitergereicht oder zu einem variablen Verzögerungsglied geleitet. Das variable Verzögerungsglied stellt den Meßauftrag zum Startzeitpunkt wieder in die Eingangswarteschlange der ersten Teilphase. Die Teilphase 3 entspricht der Rückkehr aus dem Systemaufruf zur Prozeßteilung (*fork*) und gegebenenfalls dem Start des Prozesses für die Ergebnisaufbereitung. Teilphase 4 wird nur bei Meßaufträgen mit sofortigem Start der Ergebnisaufbereitung durchlaufen. Der Zugriff auf die Jobliste wird durch die Token-Rückgabe nach Teilphase 3 oder 4 beendet.

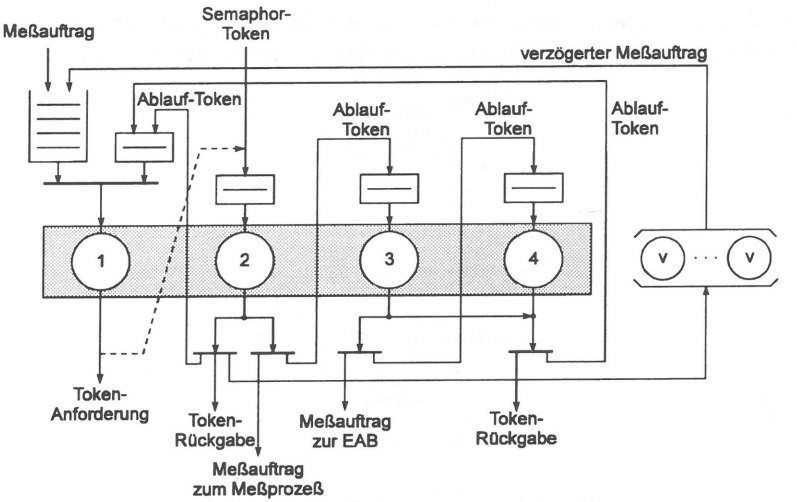


Bild 5.6: Warteschlangenmodell des Monitoring Schedulers

5.1.3.3 Kommunikationssystem

Das Kommunikationssystem des Monitoring-Systems wurde entsprechend dem Prototyp als Anwendungsprozeß modelliert, welcher zentral für alle Meßprozesse das Versenden und Empfangen von Basismeßaufträgen durchführt. Die Bearbeitung der Basismeßaufträge durch die Meßstationen wird durch eine Verzögerung der Antworten nachgebildet. Das Warteschlangenmodell des Kommunikationssystems in Verbindung mit den Meßstationen zeigt Bild 5.7.

Der Prozeß besteht aus nur einer Phase, die alle Aufträge an das Kommunikationssystem bearbeitet. Die Auftragstypen sind:

- *Open*: Anmeldung beim Kommunikationssystem,
- *Send*: Versenden eines Basismeßauftrags (dabei impliziter Verbindungsaufbau),
- *Receive*: Warten auf Antwort von der Meßstation (Receive-Req.) und Empfang derselben (Receive-Resp.),
- *Disconnect*: Abbau der Verbindung und damit Freigabe der Meßstation.

Jedem Auftragstyp ist eine individuelle Bearbeitungszeit zugeordnet. Receive-Aufträge werden nach dem ersten Durchlauf durch das Kommunikationssystem in dem variablen Verzöge-

rungsglied für die Bearbeitungsdauer der Meßstation zwischengespeichert, um danach die Phase ein zweites Mal zu durchlaufen und an den Meßprozeß zurückgesendet zu werden.

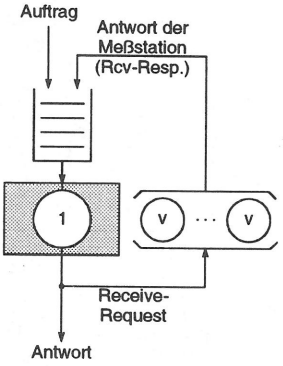


Bild 5.7: Modellierung des Kommunikationssystems und der Meßstationen

5.1.3.4 Meßdurchführung

Der reale Betrieb des Monitoring-Systems ist gekennzeichnet durch die gleichzeitige Durchführung von unterschiedlichen Meßaufträgen, die sich zwangsläufig gegenseitig beeinflussen. Die simulative Untersuchung unter verschiedenen Lastbedingungen erfordert, daß die prinzipiellen Prozeßabläufe von einmaligen und periodischen Meßaufträgen nachgebildet werden. Die Modellierung der Meßdurchführung resultiert deshalb in insgesamt 14 Teilphasen, die entsprechend dem in Bild 4.18 dargestellten Diagramm abgearbeitet werden. Nach einigen Teilphasen existieren Verzweigungspunkte, an denen, abhängig von der Meßauftragsklasse, über den weiteren Ablauf entschieden wird. Auf die Darstellung des Warteschlangenmodells soll aus Platzgründen hier verzichtet werden. Die einzelnen Teilphasen haben folgende Bedeutung:

1. Prozeßstart und Initialisierung,
2. Zugriff auf die Jobliste und Auslesen der Systemkennung des Meßjobs,
3. Auswertung der Meßparameter - Anmeldung beim Kommunikationssystem,
4. Initialisierung der Messung - Entscheidung, ob die Kennung der Message Queue der EAB aus der Jobliste ausgelesen werden muß,
5. Auslesen der Systemkennung der Message Queue der EAB aus der Jobliste,
6. Vorbereitung und Absenden eines Basismeßauftrags,
7. Vorbereitung des Empfangs des Basismeßergebnisses,
8. Auswertung des Ergebnisses - Bildung eines Zwischenergebnisses (wird evtl. mehrfach durchlaufen, falls das Ergebnis in mehreren Paketen übertragen wird),
9. Versenden des Zwischenergebnisses an die Message Queue der EAB,

10. Berechnung des Zeitpunktes für die nächste Messung bei periodischen Meßaufträgen,
11. Eintragen des Endes der Messung in die Jobliste,
12. Start der Ergebnisaufbereitung,
13. Übermittlung des Endes der Messung an die EAB,
14. Beendigung des Prozesses.

5.1.3.5 Ergebnisaufbereitung

Im Simulationsmodell der Ergebnisaufbereitung werden neben den Meßauftragsmeldungen vom Monitoring Scheduler bzw. der Meßdurchführung auch Benutzermeldungen verarbeitet, die von einem Generator innerhalb des EAB-Modells erzeugt werden. Dieser Generator bildet die Ereignisse durch Benutzeraktivitäten nach. Anzahl und Ankunftsabstand sind im Meßauftrag abgelegt und dienen zur Initialisierung des Generators während der ersten Teilphase. Meßaufträge, die bei der EAB eintreffen, haben drei Bedeutungen. Der erste Meßauftrag symbolisiert den Start der EAB durch den Monitoring Scheduler oder die Meßdurchführung. Weitere Meßaufträge entsprechen den Zwischenergebnissen, die in der Message Queue empfangen werden. Das Ende der Messung signalisiert die Meßdurchführung ebenfalls über einen Meßauftrag, bei dem ein bestimmtes Flag gesetzt ist.

Das Warteschlangenmodell der Ergebnisaufbereitung zeigt Bild 5.8. Die Teilphasen 1 bis 3 werden nur während der Initialisierung der EAB durchlaufen. Danach werden sowohl Zwischenergebnisse als auch Benutzerereignisse in Teilphase 4 verarbeitet. Teilphase 5 schließlich entspricht dem Zugriff auf die Jobliste, in dem der Meßauftrag aus den System entfernt wird.

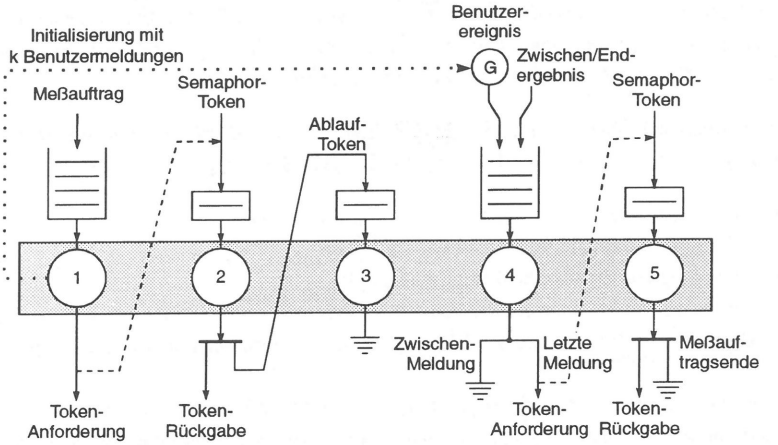


Bild 5.8: Warteschlangenmodell der Ergebnisaufbereitung

5.1.4 Prozessor

Die Ablaufsteuerung der Prozesse des Monitoring-Systems durch das Betriebssystem wird durch das Prozessormodell nachgebildet. Die Bediendauern der Teilphasen stellen reine Rechenzeiten dar. Der Tatsache, daß sich die Phasen eine Bedieneinheit teilen müssen und sich somit die Ausführungsdauern auf Prozeßebene wesentlich vergrößern, wird durch die zyklische Abarbeitung der ablauffähigen Phasen Rechnung getragen. Das Prozessormodell sieht vor, daß eine in Bearbeitung befindliche Phase durch eine höherpriore Phase unterbrochen werden kann. Der Prozessor führt verschiedene Statistiken über den prozentualen Anteil der Phasen an der gesamten Belegungszeit des Prozessors, die Wartezeiten der Phasen und den Anteil von Unterbrechungen. Es ist prinzipiell möglich, spezielle Zuteilungsstrategien zu implementieren und damit den Einfluß des Betriebssystems auf die Bedienzeiten zu untersuchen. Dies wurde jedoch unterlassen, da zum einen aufgrund der kurzen Bediendauern der Teilphasen die Wahrscheinlichkeit, unterbrochen zu werden, sehr gering ist und zum anderen die zyklische Zuteilung innerhalb einer Prioritätsstufe der Prozeßzuteilungsstrategie von UNIX sehr nahe kommt.

5.2 Umsetzung des Warteschlangenmodells in ein Simulationsprogramm

Ein Problem bei der Simulation des Monitoring-Systems liegt in der komplexen Struktur des Systems und damit des Simulationsmodells. Die Methoden der objektorientierten Programmierung erleichtern dem Implementierer, das Problem auf verschiedenen Abstraktionsebenen separat zu betrachten. In diesem Zusammenhang ist es vor allem von Vorteil, die Komponenten der Simulation von deren Verknüpfung zu einem Gesamtmodell und dem Plazieren von Meßpunkten zu trennen. Dieses Vorgehen erfährt eine besondere Unterstützung durch eine am Institut für Nachrichtenvermittlung und Datenverarbeitung entwickelte, objektorientierte Simulationsbibliothek [95], auf welche sich die hier vorgestellte Simulation stützt.

Diese Simulationsbibliothek ist eine Klassenbibliothek basierend auf der Programmiersprache C++ [24] und beinhaltet folgende Teilkomponenten eines Simulationsprogramms:

- Ablaufsteuerung,
- Führung und formatierte Ausgabe der Statistiken,
- Erzeugung von Zufallszahlen und Zufallsgrößen,
- Austausch von Meldungen zwischen den Modellkomponenten und
- Objektklassen für Standard-Modellkomponenten wie Generatoren, Warteschlangen u.a.

Alle Modellkomponenten besitzen sogenannte *Ports*, über die Meldungen ausgetauscht werden können, wobei zwischen Eingangsports und Ausgangsports unterschieden wird. Durch die Verbindung eines Ausgangsports einer Modellkomponente mit dem Eingangsport einer anderen entsteht ein unidirektionaler Kanal zur Übertragung von Meldungen. Die Erstellung eines

Simulationsprogramms beschränkt sich auf die Implementierung der in der Bibliothek nicht vorhandenen Klassen für die Modellkomponenten, der benötigten Meldungstypen und die Erzeugung eines Simulationsmodells, indem die Objekte für die Modellkomponenten erzeugt und untereinander verbunden werden.

Die Simulationsbibliothek erlaubt die Realisierung von hierarchischen Modellkomponenten; d.h. eine Komponente, die bestimmte externe Ports besitzt, kann wiederum aus Modellkomponenten bestehen, die untereinander und mit den externen Ports verbunden sind. Dadurch kann die Hierarchie des Simulationsmodells des Monitoring-Systems direkt in eine Objekthierarchie umgesetzt werden.

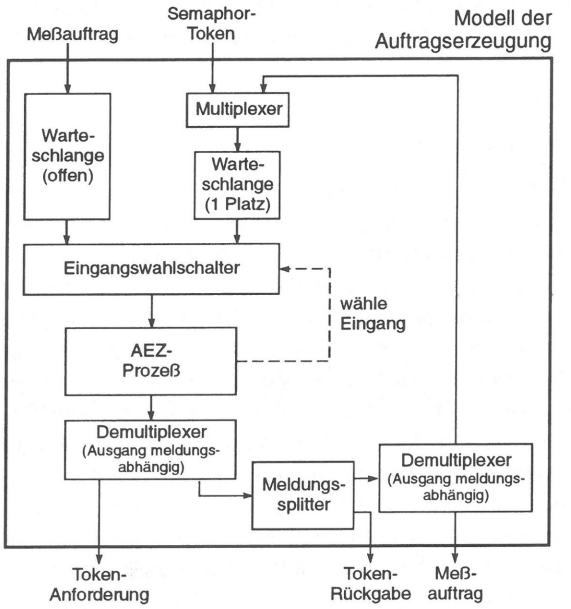


Bild 5.9: Realisierung der Auftragserzeugung in einem Simulationsprogramm

Diese Umsetzung ist in Bild 5.9 anhand der Auftragserzeugung verdeutlicht. Eine deutliche Reduktion der Komplexität leistet die Zusammenfassung aller Teilphasen eines Prozesses zu einer Modellkomponente, hier mit dem Namen *AEZ-Prozeß*. Diese Komponente präsentiert sich dem Prozessor als eine Phase mit einer internen Zustandsvariablen zur Speicherung der aktuellen Teilphase, also der Position im Programmablauf. Die Bedienzeit dieser Phase ist abhängig vom internen Zustand, d.h. der aktuellen Teilphase. Der AEZ-Prozeß gewährleistet den korrekten Ablauf der Teilphasen und stellt über den Eingangswahlschalter den Empfang der

richtigen Meldungen sicher. Dadurch können die in den Warteschlangenmodellen enthaltenen Synchronisationselemente an den Eingängen der Teilphasen entfallen. Die Modellkomponenten der übrigen Prozesse des Monitoring-Systems sind ähnlich aufgebaut und unterscheiden sich hauptsächlich in den möglichen Wegen der Meldungen am Ausgang der Prozeßkomponente.

5.3 Simulative Leistungsuntersuchung des Monitoring-Systems

Die Leistungsuntersuchung eines Systems mit Hilfe der stationären Simulation stellt ein experimentelles Verfahren dar, bei dem über zufällig erzeugte Eingangsereignisse ein stationärer Betriebszustand des Systems eingestellt wird und die interessierenden Leistungskenngrößen durch statistische Messungen erfaßt werden. Die Ergebnisse sind Stichproben, deren Aussagesicherheit mit den Methoden der beurteilenden Statistik in Form von Vertrauensintervallen berechnet wird.

5.3.1 Simulationsparameter

Mit der Komplexität eines Simulationsmodells wächst auch die Freiheit in der Wahl der Simulationsparameter, welche die Leistungsfähigkeit des Systems beeinflussen. Werden von der Simulation realistische Ergebnisse erwartet, müssen diese Parameter so gewählt werden, daß damit das System möglichst exakt nachgebildet wird. Bei der Simulation des Monitoring-Systems sind dies vor allem die Bedienzeiten der Teilphasen, zu deren Bestimmung an der prototypischen Implementierung verschiedene Messungen durchgeführt wurden. Diese Messungen liefern die Mittelwerte und die Schwankungsbreiten der im übrigen gleichverteilt angenommenen Bediendauern. Da die Rechenzeiten der einzelnen Prozesse des Monitoring-Systems das zeitliche Auflösungsvermögen des Betriebssystems unterschritten, mußten diese, analog zu den Leistungsuntersuchungen der IPC-Mechanismen in Kapitel 4.1.5, auf einem im übrigen freien Rechnersystem wiederholt ausgeführt und die Gesamtzeit gemessen werden. Auf diese Weise ist es jedoch unmöglich, die Rechenzeiten der Teilaktivitäten eines Prozesses zu bestimmen. Deshalb wurden in den Programmen systematisch durch bedingte Übersetzung bestimmte Programmabschnitte herausgenommen bzw. unterschiedliche Meßauftragsszenarien untersucht und die Laufzeiten verglichen. Die Messungen lieferten folgende Werte:

- Prozeßstart 35 ms,
- Analyse eines Meßauftrages (mit Konfigurationsserver) 5-15 ms,
- einmalige Anfrage beim Konfigurationsserver 1-2 ms,
- Erzeugung eines Meßauftrages 6 ms.

Aus diesen Werten ergeben sich die in Tabelle 5.1 angegebenen Bediendauern für das Simulationsmodell.

Nummer der Teilphase	1	2	3	4
Bedienzeit in ms	40-50	0,5	3-7	0,5

Tabelle 5.1: Bedienzeiten der Teilphasen der Auftragserzeugung

Da der Monitoring Scheduler die meiste Zeit in einem Wartezustand verbringt, muß für die Bestimmung der Rechenzeiten auf die vom Betriebssystem abfragbaren Prozessorbelegungszeiten (CPU-Zeiten) eines Prozesses zurückgegriffen werden. Der Monitoring Scheduler wurde so erweitert, daß er über ein Signal dazu veranlaßt werden kann, die eigene CPU-Zeit und die Summe der CPU-Zeiten der beendeten Kindprozesse (Meßdurchführung und Ergebnisaufbereitung) auszugeben. Aufgrund der gemessenen Zeiten für die Einplanung von Meßaufträgen, den alleinigen Start der Meßprozesse und die gemeinsame Erzeugung von Meßprozeß und Ergebnisaufbereitung, ergaben sich die in Tabelle 5.2 aufgeführten Werte.

Nummer der Teilphase	1	2	3	4
Bedienzeit in ms	0,25	0,25-1,25	1-2	1-2

Tabelle 5.2: Bedienzeiten der Teilphasen des Monitoring Schedulers

Auch das Kommunikationssystem existiert ständig im Hintergrund. Die Bedienzeiten aus Tabelle 5.3 wurden anhand einmaliger und periodischer Meßaufträge, sowie Aufträgen mit mehreren Antworten auf einen Basismeßauftrag abgeschätzt.

Auftragstyp	Open	Send	Rcv.-Req.	Rcv.-Resp.	Disconnect
Bedienzeit in ms	0,5	0,75-1,25	0,5	1-2	0,5

Tabelle 5.3: Bedienzeiten des Kommunikationssystems

Die Aufteilung auf die Bedienzeiten der Teilphasen der Meßdurchführung kann aus Tabelle 5.4 entnommen werden.

Nummer der Teilphase	1	2	3	4	5	6	7
Bedienzeit in ms	15-25	0,5	1-2	1-2	0,5	15-25	5-15

Nummer der Teilphase	8	9	10	11	12	13	14
Bedienzeit in ms	100-200	10	15-25	0,5	1-2	2	0,5

Tabelle 5.4: Bedienzeiten der Meßdurchführung

Die ersten beiden Teilphasen der Ergebnisaufbereitung entsprechen denen der Meßdurchführung, so daß deren Ausführungszeiten übernommen wurde. Bei Teilphase 3 entfällt die Anmel-

dung beim Kommunikationssystem, wodurch sie etwas kürzer ausfällt. Die Ausführungszeiten für die Teilphasen 4 und 5 entspringen aufgrund mangelnder Meßmöglichkeiten groben Schätzungen.

Nummer der Teilphase	1	2	3	4	5
Bedienzeit in ms	15-25	0,5	0,5	5-25	1

Tabelle 5.5: Bedienzeiten der Ergebnisaufbereitung

Neben den Bedienzeiten der Teilphasen der Prozesse des Monitoring-Systems ist die Konstellation der Meßaufträge und ihre Ankunftsverteilung für die Simulation festzulegen. Unter Annahme der Gedächnisfreiheit wurden negativ-exponentiell verteilte Ankunftsabstände gewählt.

5.3.2 Verzögerung von einmaligen Meßaufträgen

Ein einmaliger Meßauftrag wird an zwei Stellen im Monitoring-System verzögert, und zwar bei der Generierung und beim Start der Messung. Bei Meßaufträgen, die nicht sofort beginnen, fällt die Verzögerung durch die Generierung nicht ins Gewicht, so lange dadurch der Startzeitpunkt der Messung nicht überschritten wird.

Das Zeitverhalten einmaliger Meßaufträge wurde an drei verschiedenen Meßauftragstypen untersucht, die sich in der Verzögerung des Startzeitpunktes unterscheiden:

- Auftrag 1: Start sofort,
- Auftrag 2: Start nach 0 - 200 ms (gleichverteilt),
- Auftrag 3: Start nach genau 1 s.

Die Prozessorauslastung wurde über den Ankunftsabstand der Meßaufträge eingestellt. Die übrigen Auftragsparameter sind:

- Startzeitpunkt der EAB: am Ende der Messung,
- Messungen: 1 Basismeßauftrag mit 1 Antwort nach 100 ms
- Benutzerverhalten: 1 Ereignis nach 500-800 ms (gleichverteilt).

Bild 5.10 zeigt die Mittelwerte der Startverzögerung, während Bild 5.11 die 95%-Quantile (Wert, der 95% der Ereignisse nach oben begrenzt) darstellt. In diesen und den folgenden Diagrammen sind Vertrauensintervalle zu einer statistischen Sicherheit von 95% eingezeichnet, falls sie nennenswert sind. Die Startverzögerung bezeichnet die Dauer zwischen dem Startzeitpunkt der Messung und dem Ende der Auswertung der Auftragsparameter durch die Meßdurchführung. Man erkennt, daß für die Auftragsklasse 2 bei niedriger Prozessorauslastung die

Bearbeitungszeit durch die AEZ nicht zum tragen kommt, während sich die Kurven bei höherer Last denen der Auftragsklasse 1 nähern.

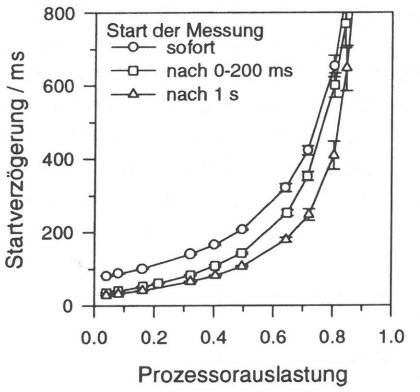


Bild 5.10: Mittelwerte der Startverzögerung bei kurzen Messungen

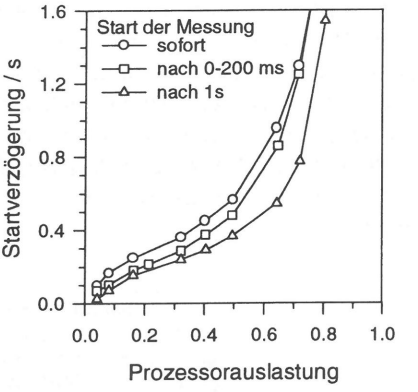


Bild 5.11: 95%-Quantile der Startverzögerung bei kurzen Messungen

In Bild 5.12 ist die Dauer einer Messung, gemessen vom Beginn der Bearbeitung durch den Monitoring Scheduler bis zum Ende des Meßprozesses, über der Prozessorauslastung aufgetragen.

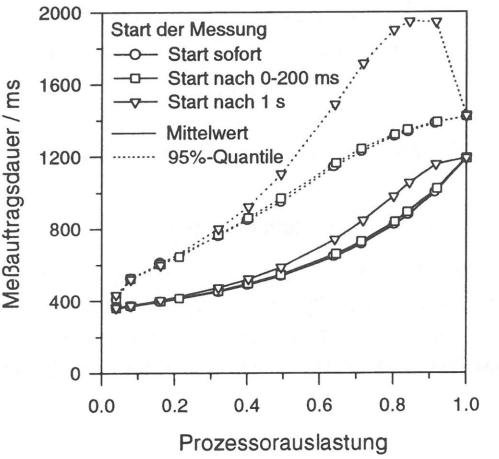


Bild 5.12: Meßauftragsdauer bei kurzen Messungen

Die längeren Ausführungszeiten der Auftragsklasse 3 bei höheren Auslastungen sind darauf zurückzuführen, daß sich im System auch wartende Meßaufträge befinden, die unabhängig von den gerade erzeugten Meßaufträgen zusätzlich zur Bearbeitung anstehen können. Bei nicht verzögerten Messungen ist die mittlere Wartezeit vor dem Monitoring Scheduler gleich null, da dieser Prozeß aufgrund der Prozeßzuteilungsstrategie immer nach der AEZ zugeteilt wird. Es gilt jedoch generell, daß bei einem Wartesystem ein Ankunftsprozeß mit geringerer Varianz eine kürzere Wartezeit zur Folge hat [93]. Befindet sich das System in Überlast, bildet bei kurzen Messungen die Auftragserzeugung den Engpaß. Das bedeutet wiederum, daß bei allen Meßaufträgen der Startzeitpunkt überschritten ist, wodurch sich der Abfall der Kurven für den Auftragstyp 3 erklären läßt. Dies wird durch die Bild 5.13 dargestellte Belegung der Jobliste bestätigt.

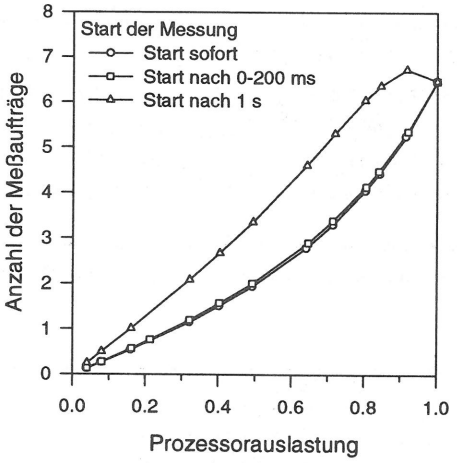


Bild 5.13: Mittlere Anzahl von Meßaufträgen in der Jobliste

5.3.3 Verhalten bei periodischen Meßaufträgen

Eine zweite Studie sollte klären, inwieweit einzelne Messungen bei periodischen Meßaufträgen, abhängig von der Auslastung des Systems, verzögert werden. Hierzu wurde die Verzögerung einer Teilmessung, definiert als die Differenz zwischen der theoretischen Startzeit eines Basismeßauftrags und dem tatsächlichen Startzeitpunkt, bei drei Meßauftragskonstellationen gemessen:

- Meßreihe 1: periodische Meßaufträge mit langen Teilmessungen (je 20 Teilmessungen, alle 5 s - eine Teilmessung bestehend aus 3 Basismeßergebnissen je 100 ms verzögert),
- Meßreihe 2: periodische Meßaufträge mit kurzen Teilmessungen (wie Meßreihe 1, jedoch mit nur 1 Basismeßergebnis),
- Meßreihe 3: eine Kombination aus periodischen Meßaufträgen entsprechend Meßreihe 2 mit einer gleichverteilten Ankunftsrate von 10 - 30 s und zusätzlich einmaligen Meßaufträgen entsprechend Abschnitt 5.3.2 mit variabler Ankunftsrate.

Die Prozessorauslastung wurde bei den Meßreihen 1 und 2 über die mittlere Auftragsrate eingestellt, die wiederum die Anzahl der parallelen Meßaufträge bestimmt. Für die Meßreihe 3 entspricht die Grundlast durch die Hintergrundmessungen einer Prozessorauslastung von 24%, ihr Ankunftsabstand ist in einem relativ engen Intervall gleichverteilt, um Schwingungen des Systems zu vermeiden.

Bild 5.14 zeigt die mittleren Verzögerungszeiten der Teilmessungen und deren 95%-Quantile für die drei Meßreihen im Vergleich. Es zeigt sich, daß die Kurven bis zu einer für die Meßreihe spezifischen kritischen Prozessorauslastung relativ flach verlaufen, um danach stark anzusteigen. Die kritische Grenze der Prozessorauslastung gilt sowohl für den Mittelwert, als auch das 95%-Quantil der Verzögerungszeit.

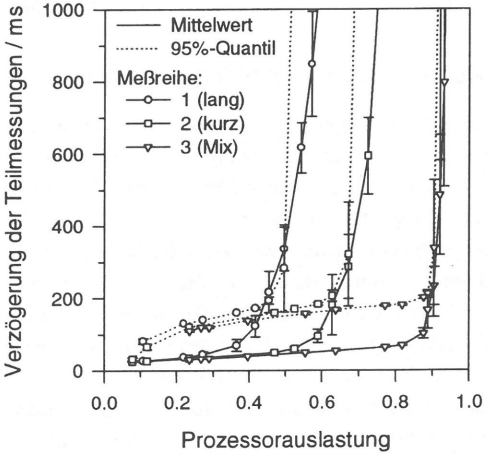


Bild 5.14: Verzögerung der Teilmessungen bei periodischen Meßaufträgen

Die Meßreihe 3 zeigt, daß einmalige Meßaufträge periodische Hintergrundmessungen bis zu einer relativ hohen Prozessorauslastung nur wenig stören. Erst wenn, wie aus Bild 5.15 ersichtlich, die mittlere Anzahl der Meßaufträge in der Jobliste stark ansteigt und die einmaligen Meßaufträge, wie in Bild 5.16 dargestellt, sich gegenseitig behindern, steigt auch die Verzögerung der Hintergrundmessungen.

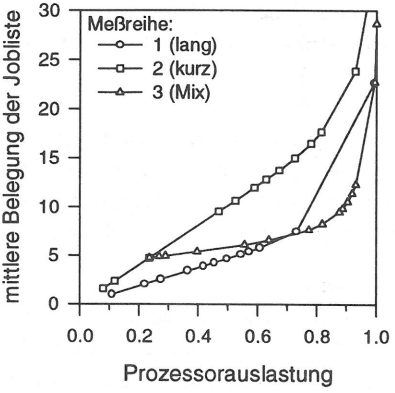


Bild 5.15: Mittlere Anzahl von Meßaufträgen in der Jobliste bei periodischen Messungen

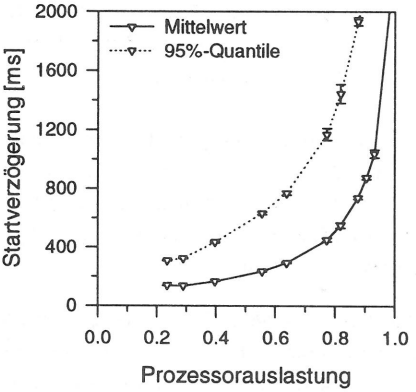


Bild 5.16: Verzögerung der einmaligen Meßaufträge bei periodischen Hintergrundmessungen

5.4 Zusammenfassung der Simulationsergebnisse

Die durchgeführten Simulationen zeigen, daß die kritische Schwelle der Prozessorauslastung von der Zusammensetzung der Meßaufträge abhängt und in einem Bereich von 40 % bis 80% schwankt. Für das betrachtete Rechnersystem bildete die lange Bearbeitungsdauer einer Messung den Flaschenhals, und begrenzte die Anzahl der gleichzeitig durchführbaren Messungen. Über die Änderung der Systemarchitektur bzw. eine Priorisierung der Prozesse werden keine wesentlichen Verbesserungen erwartet, da die zentralen Elemente keinen Engpaß bildeten. So lag z.B. die mittlere Wartezeit auf ein Token des Semaphors wesentlich unter der Bearbeitungsdauer eines Basismeßauftrags bei der Meßdurchführung. Trotz der einfachen Prozessorzuteilungsstrategie wurden akzeptable Werte für die 95%-Quantilen der Verzögerungszeiten erreicht. Interessant ist auch die Feststellung, daß sich für die Kombination von Hintergrundmessungen und einmaligen Meßaufträgen ein sehr gutes Leistungsverhalten erzielen ließ.

Kapitel 6

Zusammenfassung und Ausblick

Die eigentliche Problematik bei der Überwachung von Rechnernetzen besteht darin, eine Fülle von im Netz verteilt erfaßten Daten an einer zentralen Stelle zu sammeln und so zu verarbeiten, daß dem Anwender nur die für ihn relevanten Informationen präsentiert werden. Entscheidend dabei ist, daß nicht von vornherein festgelegt werden kann, welche Informationen relevant sind. Dies steht im Gegensatz zu der Überwachung vieler anderer Systeme.

Im Rahmen der vorliegenden Arbeit wurden, nach einer Einführung in die Netzmanagementkonzepte, die Grundlagen dargestellt, auf denen die Überwachung von Rechnernetzen basiert. Dabei wurden die zum Einsatz kommenden Meßsysteme klassifiziert, ihre Meßprinzipien kurz beschrieben und die Kommunikationsmechanismen und Protokolle zur Übertragung der erfaßten Daten an eine zentrale Station vorgestellt. Die zwei dominierenden Standards für das Netzmanagement (OSI und SNMP) wurden im Hinblick auf ihre Überwachungsaspekte eingehend beleuchtet. Es zeigte sich dabei, daß viele der verfügbaren verteilten Meßsysteme nicht über das standardisierte Netzmanagement bedienbar sind.

In Kapitel 3 wurde ein neues Konzept vorgestellt, welches die einheitliche Bedienung und Verwaltung verschiedenartiger Meßsysteme erlaubt. Dieses Konzept zeichnet sich dadurch aus, daß die mit den Meßsystemen durchgeführten Messungen in den Vordergrund gestellt werden, statt, wie für das Netzmanagement üblich, sich auf die erfaßten Daten zu konzentrieren. Die Umsetzung dieses Konzept resultierte in der Entwicklung des Monitoring-Systems, welches in mehrfacher Hinsicht offen und universell ist:

- Verschiedene Meßsysteme können leicht integriert werden.
- Die durchführbaren Messungen sind nicht vordefiniert.
- Das System kann leicht um neue Meßsysteme und Messungen erweitert werden.
- Durch eine entsprechende Anpassung der Benutzungsoberfläche kann verschiedenen Anwendertypen entsprochen werden.

Das Monitoring-System unterstützt den Anwender bei der zentralen Organisation und Durchführung der Messungen. Die Dreiteilung einer Messung in Auftragserzeugung, Meßdurchführung und Ergebnisaufbereitung schafft den Rahmen, in welchem Meßfunktionen, Meßsysteme sowie die Ergebnisaufbereitung und -präsentation verschiedenartig kombiniert werden können. Die klaren und einfachen Schnittstellen zwischen dem Kern des Monitoring-Systems und der Realisierung der Benutzungsoberfläche erlauben, das System in unterschiedlichen Bereichen einzusetzen, bzw. auf verschiedene Arten in eine Netzmanagementarchitektur einfügen.

Die Nebenläufigkeit der Messungen und die geforderte Flexibilität führte dazu, das System als ein kooperierendes Mehrprozeßsystem zu konzipieren. Die Auswahl der Kommunikationsmechanismen zwischen den Prozessen wurde mit großer Sorgfalt getroffen, damit das System aufgrund seiner Prozeßarchitektur den zeitlichen Anforderungen gerecht werden kann. Hierzu gehört auch die Realisierung eines schnellen Zugriffs auf Konfigurationsdaten des Netzes und der Meßsysteme durch einen speziellen Serverprozeß, ohne die Flexibilität eines Datenbanksystems zu verlieren. Die Integration der Meßfunktionen und Darstellungsarten werden für den Anwender dadurch erreicht, daß eine speziell entwickelte Beschreibungssprache für Meßaufträge zur Verfügung steht, mit der alle Parameter der Messung beschrieben werden können.

Die Realisierung durch den Prototyp konzentrierte sich auf die Ablaufsteuerung der Meßaufträge und die allgemeine Infrastruktur zur Durchführung der Messungen. Bei den implementierten Meßfunktionen sind vor allem die automatischen Reflektionsmessungen während des Netzbetriebs zu erwähnen, bei denen die Reflektionskurven sofort angezeigt werden können und somit erstmalig der wirkliche Einsatz dieser Meßkomponente für die Netzüberwachung ermöglicht wird.

Die Frage, wie sich das Monitoring-System im Einsatz in größeren Netzen verhalten wird, kann experimentell auch bei der Verfügbarkeit eines Prototypen nur sehr schwer beantwortet werden. Wird das System verkehrstheoretisch modelliert, ist es in simulativen Untersuchungen möglich, die Leistungsfähigkeit des Systems in verschiedenen Betriebszuständen vorherzusagen. Die durchgeführten Simulationen ergaben, daß die Leistungsgrenze des Monitoring-Systems auf der betrachteten Rechnerplattform der begrenzten Rechenkapazität bei der Meßdurchführung zuzuschreiben ist und durch die Architektur nicht wesentlich verbessert werden kann. Die Prozessorzuteilungsstrategie scheint relativ unkritisch zu sein. Es konnte gezeigt werden, daß die Beeinflussung von Hintergrundmessungen durch sporadische Meßaufträge gering ist und das System in dieser Kombination hoch ausgelastet werden kann.

Die Entwicklung des Monitoring-Systems ist ein Schritt in die Richtung, wichtige Funktionalitäten des Netzmanagements auf der Seite des Managers zu realisieren, um so dem Netzadministrator weitergehende Methoden als die Abfrage von Netzinformationen über standardisierte Protokolle und Datenmodelle in die Hand zu geben. Das Monitoring-System befreit den Anwender jedoch nicht davon, darüber nachzudenken, welche Messungen er für die Überwa-

chung seines Netzes im Detail benötigt. Es schafft lediglich einen Rahmen, in dem der zeitliche Ablauf der Messungen koordiniert wird und die Übersicht über die Messungen und die Meßsysteme gewahrt bleibt. Es hat sich gezeigt, das die Konzepte, die dem Monitoring-System zugrunde liegen, nicht auf die Netzüberwachung beschränkt sind. Vielmehr ist ein System entstanden, welches ganz allgemein in der Lage ist, Vorgänge (Messungen) zeitlich zu organisieren und die Ergebnisse dieser Vorgänge vielfältig zu präsentieren. Für den Anwender werden diese Vorgänge als Objekte modelliert, auf die er verschiedene Operationen ausführen kann.

Der Aufwand, mehrere Meßfunktionen für verschiedene Meßsysteme zu implementieren, ist trotz allgemeiner Module immer noch beträchtlich. Hinzu kommen die verschiedenen Aufbereitungsfunktionen für die Ergebnisse. Hier könnte mit automatischen Software-Generierungsmethoden, bei Vorgabe der externen Schnittstellen, wie der Parameterliste im Meßjob, den erforderlichen Szenarien der Basismeßaufträge und des Formats des Ergebnisteils, die Erzeugung der Meßprogramme unterstützt werden. Dasselbe Prinzip könnte auch bei der Ergebnisaufbereitung Anwendung finden.

Bei sternförmig aufgebauten Netzen oder Hochgeschwindigkeitsnetzen werden die Überwachungsfunktionen stärker in die Netzkomponenten verlagert. Dies führt zu einem starken Rückgang der Anzahl der verteilten Meßsysteme im Netz. Da diese Netzkomponenten bereits mit Netzmanagementschnittstellen ausgestattet sind, werden die dort angesiedelten Überwachungsfunktionen ebenfalls darüber zugänglich sein und sind dem Netzmanagement direkt zugänglich. Trotzdem werden jedoch weiterhin spezielle Systeme wie das Monitoring-System für die netzweite Überwachung ihren Stellenwert beibehalten, da Netzmanagementplattformen die besonderen zeitlichen Anforderungen der Überwachung in der Regel nicht erfüllen.

Literaturverzeichnis

[1] ANSI X3.229: *Information Systems – Fibre Distributed Data interface (FDDI) – Station Management (SMT)*, American National Standard, 1994.

[2] van As, H.; Lemppenau, W.; Schindler, H.; Zafiropulo, P.: *CRMA-II: A MAC protocol for ring-based Gb/s LANs and MANs*, Computer Networks and ISDN Systems, Vol. 26, 1994, Seite 831-840.

[3] Auer, S.: *Spezifikation einer Beschreibungssprache für einen Meßauftrag des Monitoringsystems und Implementierung eines Interpreters*, Diplomarbeit Nr. 1147, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992.

[4] Bach, M.: *The Design of the Unix Operating System*, Prentice-Hall International, London, 1986.

[5] Bachmann, P.: *Grundlagen der Compilertechnik, Verfahren der Datenverarbeitung*, R. Oldenbourg Verlag, München, 1975.

[6] Batini, C.; Ceri, S.; Navathe, S.: *Conceptual Database Design, An Entity-Relationship Approach*, Benjamnin/Cummings Publishing Company, Redwood City, California, 1992.

[7] Bayreuther, M.: *Entwurf und Implementierung der SNMP-Task für den Internet Proxy Agent*, 2. Semesterarbeit Nr. 1141, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992.

[8] Bönsch, W.: *Aufbau und Integration eines automatischen Reflexionsmeßgerätes für LAN Segmente nach IEEE 802.3*, Diplomarbeit Nr. 1051, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990.

[9] Booch, G.: *Object-Oriented Analysis and Design*, Second Edition, Benjamin/Cummings Publishing Company, Redwood City, California, 1994.

[10] Bosch, M.; Rößler, G.; Schollenberger, W.: *Network Management in Heterogeneous Networks for Factory Automation*, Proceedings Information Network and Data Communication (INDC), Lillehammer, Norwegen, 1990, Paper 1A/2.

[11] Brinkmann, W.: *Distributed Sniffer System (DSS) von Network General*, Datacom, Vol. 11, Nr. 10/11, Oktober/November 1994, Seite 152-156/154-161.

[12] Case, J.; Fedor, M.; Schoffstall, M.; Davin, J.: *A Simple Network Management Protocol (SNMP)*, RFC 1157, Mai 1990.

[13] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S.: *Introduction to version 2 of the Internet-standard Network Management Framework*, RFC 1441, März 1993.

[14] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S.: *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)*, RFC 1142, März 1993.

[15] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S.: *Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)*, RFC 1450, März 1993.

[16] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S.: *Manager-to-Manager Management Information Base*, RFC 1451, März 1993.

[17] Chen, P.: *The Entity-Relationship Model – Towards a Unified View of Data*, ACM Transactions on Database Systems, Vol. 1, Nr. 1, März 1976, Seite 9-36.

[18] Coffield, D.; Shepherd, D.: *Tutorial guide to Unix sockets for network communications*, Computer Communications, Vol. 10, Nr. 1, Februar 1987, Seite 21-29.

[19] Comer, D.: *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture*, Prentice-Hall International, London, 1991.

[20] Datapro Research: *Network Management Functions – Network Monitoring and Control*, in: Datapro Network Management, McGraw-Hill, Delran, NJ, 1989.

[21] Daubner, J.: *Entwicklung einer Konfigurationsdatenbank für das Monitoringsystem*, Diplomarbeit Nr. 1245, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1993.

[22] Digital Equipment Corporation, Intel, Xerox: *The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications*, Version 1.0, September 1980.

[23] DIN 66253: *Die Programmiersprache PEARL*, Deutsches Institut für Normung, Beutz-Verlag, Berlin, 1980.

[24] Ellis, M.; Stroustrup, B.: *The Annotated C++ Reference Manual*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.

[25] ESPRIT Project 7096 CCE-CNMA: *CNMA-Implementation Guide Revision 6.01*, Esprit Project „CCE-CNMA" Management Committee, Juli 1993.

[26] Gaffin, A.: *Everybody's guide to the Internet*, M.I.T. Press, 1994.

[27] Gerblinger, D.: *Integration eines Kommunikationssystems für ein verteiltes LAN-Meßsystem*, Diplomarbeit Nr. 1149, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992.

[28] Gihr, O.: *Analyse datenflußregulierter Verbindungskonzepte in verteilten Systemen mit mehrschichtiger Protokollarchitektur*, 48. Bericht über verkehrstheoretische Arbeiten, Dissertation, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990.

[29] Gulbins, J.: *UNIX – Eine Einführung in Begriffe und Kommandos*, Springer-Verlag, Berlin, 1988.

[30] Grehan, R.: *Just Between Friends: Talking Tasks*, Byte, Vol. 15, No. 10/12, Oktober/November 1990, Seite 311-320/403-416.

[31] Harjung, R.: *Entwicklung einer Testumgebung für das Simple Network Management Protocol (SNMP)*, 2. Semesterarbeit Nr. 1122, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1991.

[32] Hegering, H.: *OSI-Netzmanagement*, in: H. Kerner: Rechnernetze nach OSI, Addison-Wesley Publishing Company, Bonn, 1992.

[33] Hegering, H.; Abeck, S.; Valta, R.: *Integriertes Netzmanagement – Konzepte und Realisierungen*, Tutorium Kommunikation in Verteilten Systemen (KIVS), München, März 1993.

[34] Herrtwich, R; Hommel, G.: *Kooperation und Konkurrenz – Nebenläufige, verteilte und echtzeitabhängige Programmsysteme*, Studienreihe Informatik, Springer-Verlag, Berlin, 1989.

[35] Hewlett Packard: *HP EtherTwist Hub and Hub Plus*, Hewlett Packard, 1992.

[36] Hewlett Packard: *HP Lanprobe II*, Hewlett Packard, 1992.

[37] Hewlett Packard: *HP OpenView Network Node Manager*, Hewlett Packard, 1993.

[38] Hewlett Packard: *HP OpenView Platform Developer Courses*, Hewlett Packard, 1992.

[39] Hewlett Packard: *HP OpenView Probe Manager*, Hewlett Packard, 1992.

[40] Hewlett Packard: *HP OpenView: SNMP Programmer's Guide and Reference*, Hewlett Packard, 1992.

[41] Hewlett Packard: *HP J2535A Traffic Probe, HP EASE Sampling Device for Ethernet/802.3 10Base-T, 10Base2*, Hewlett Packard, 1993.

[42] Hewlett Packard: *HP9000 Series 300/400 and 600/700/800 Computer: Berkeley IPC Programmer's Guide*, Hewlett Packard, 1991.

[43] Hewlett Packard: *HP9000 Series 300/400 and 600/700/800 Computer: LLA Programmer's Guide*, Hewlett Packard, 1991.

[44] Hewlett Packard: *HP9000 Series 300/400 and 600/700/800 Computer: NetIPC Programmer's Guide*, Hewlett Packard, 1991.

[45] Höns, G.; Köpke, F.: *Warum manche Filetransfers langsamer sind*, Datacom, Vol. 9, Nr. 9, September 1992, Seite 100-104.

[46] Hunter, B.; et. al.: Surveying Far-Flung Networks, Byte, Vol. 17, Nr. 8, August 1992, Seite 204-220.

[47] IBM: *Steuerbarer Ringleitungsverteiler IBM 8230: Sichere Token-Ring Netze*, Produktinformationen, IBM.

[48] ISO/IEC 7498: *Information Processing Systems – Open System Interconnection – Basic Reference Model*, International Standard, 1984.

[49] ISO/IEC 7498-4: *Information Processing Systems – Open System Interconnection – Basic Reference Model – Part 4: Management Framework*, International Standard, 1984.

[50] ISO 8571: *Information Processing Systems – Open System Interconnection – File Transfer, Access and Management*, International Standard, 1988.

[51] ISO 8649: *Information Processing Systems – Open System Interconnection – Service Definition for the Association Control Service Element*, International Standard, 1988.

[52] ISO/IEC 8802-2: *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical link control*, International Standard, 1994.

[53] ISO/IEC 8802-3: *Information technology – Local and metropolitan area networks – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, International Standard, 1993.

[54] ISO/IEC 8802-4: *Information processing systems – Local area networks – Part 4: Token-passing bus access method and physical layer specifications*, International Standard, 1990.

[55] ISO/IEC 8802-5: *Information Processing Systems – Data Communications – Local Area Networks – Part 5: Token Ring Access Method and Physical Layer Specifications*, International Standard, 1992.

[56] ISO/IEC 8802-6: *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements– Part 6: Distributed Queue Dual Bus (DQDB) access method and physical layer specifications*, International Standard, 1994.

[57] ISO/IEC 9072: *Information Processing Systems – Text Communication – Remote Operations*, International Standard, 1989.

[58] ISO 9314: *Information processing systems – Fibre Distributed Data Interface (FDDI)*, International Standard, 1989.

[59] ISO/IEC 9506-1: *Industrial Automation Systems – Manufacturing Message Specification – Part 1: Service Definition*, International Standard, 1990.

[60] ISO/IEC 9595: *Information Processing Systems – Open System Interconnection – Common Management Information Service Definition*, International Standard, 1991.

[61] ISO/IEC 9596-1: *Information Processing Systems – Open System Interconnection – Common Management Information Protocol – Part 1: Specification*, International Standard, 1991.

[62] ISO/IEC 10040: *Information Processing Systems – Open System Interconnection – Systems management overview*, International Standard, 1992.

[63] ISO/IEC 10164: *Information Processing Systems – Open System Interconnection – Systems Management*, Series of International Standards/Draft International Standards.

[64] ISO/IEC 10164-4: *Information Processing Systems – Open System Interconnection – Systems Management: Alarm Reporting Function*, International Standard, 1992.

[65] ISO/IEC 10164-5: *Information Processing Systems – Open System Interconnection – Systems Management: Event Report Management Function*, International Standard, 1993.

[66] ISO/IEC 10164-6: *Information Processing Systems – Open System Interconnection – Systems Management: Log Control Function*, International Standard, 1993.

[67] ISO/IEC 10164-7: *Information Processing Systems – Open System Interconnection – Systems Management: Security alarm reporting function*, International Standard, 1992.

[68] ISO/IEC 10164-11: *Information Processing Systems – Open System Interconnection – Systems Management: Metric objects and attributes*, International Standard, 1994.

[69] ISO/IEC 10164-12: *Information Processing Systems – Open System Interconnection – Systems Management: Test Management Function* , International Standard, 1994.

[70] ISO/IEC 10164-13: *Information Processing Systems – Open System Interconnection – Systems Management: Summarization Function*, International Standard, 1995.

[71] ISO/IEC 10165-1: *Information Processing Systems – Open System Interconnection – Management Information Services – Structure of management information: Management Information Model*, International Standard, 1993.

[72] ISO/IEC 10165-4: *Information Processing Systems – Open System Interconnection – Management Information Services – Structure of management information – Part 4: Guidelines for the definition of managed objects*, International Standard, 1992.

[73] ITU-T Recommendation M.3010: *Principles for a Telecommunications Management Network*, 1993.

[74] ITU-T Recommendation M.3100: *Generic Network Information Model*, 1993.

[75] ITU-T Recommendation Q.750: *Overview of Signalling System No. 7 Management*, 1994.

[76] ITU-T Recommendation X.208: *Specification of Abstract Syntax Notation One (ASN.1)*, 1989.

[77] ITU-T Recommendation X.500: *Information Technology – Open System Interconnection – The directory: Overview of concepts, models and services*, 1995.

[78] Jakubasch, B.: *Entwurf und Implementierung eines Software-Werkzeugs zur automatischen Überwachung der Netzkonfiguration auf der Vermittlungsschicht*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1995.

[79] Jander, M.: *A Protocol Analyzer With a Mind of Its Own*, Data Communications International, Vol. 20, Nr. 5, April 1991, Seite 135-136.

[80] Janssen, R.: *DME Alea Jacta Est*, Datacom, Vol. 8, Nr. 12, Dezember 1991.

[81] Janssen, R.; Schott, W.: *SNMP - Das Phänomen*, Datacom, Vol. 18, Nr. 9/10, September/Oktober 1993, Seite 150-156/157-161.

[82] Jensen, K.; Wirth, N.: *Pascal User Manual and Report*, 3rd Edition, Springer-Verlag, New York, 1985.

[83] Kastenholz, F.: *Definitions of Managed Objects for the Ethernet-like Interface Types*. RFC 1643, Juli 1994.

[84] Kauffels, F.: *Netzwerk-Management – Einführende Bestandsaufnahme und Ausblick Teil 4: Netzmanagement in SNA, grundsätzliche Struktur*, Datacom, Vol. 6, Nr. 6, Juni 1989, Seite 82-88.

[85] Kauffels, F.: *Netzwerk-Management – Einführende Bestandsaufnahme und Ausblick Teil 8: DEC – Enterprise Management Architecture EMA*, Datacom, Vol. 7, Nr. 1, Januar 1990, Seite 82-86.

[86] Kehoe, B.: *Zen & the Art of Internet*, Prentice-Hall, 1994

[87] Kell, H.; Moayeri, B.; Suppan-Borowka, J.: *ISOLAN Test Unit 1105*, Datacom, Vol. 8, Nr. 10, Oktober 1989, Seite 22-24.

[88] Kell, H.; Mues, D.; Suppan, J.: *Ethernet-Box und NET-Control*, Datacom, Vol. 7, Nr. 10, Oktober 1990, Seite 34-44.

[89] Kemmler, A: *Software für LAN-Analyse und Diagnose in Fernwirk- und Leitstellensystemen*, Diplomarbeit Nr. 1246, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1993.

[90] Kempf, R.: *Entwurf und Implementierung einer Verkehrsmeßkomponente für ein verteiltes LAN-Meßsystem*, 2. Semesterarbeit Nr. 1118, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1991.

[91] Keusch; R.: *Portierung einer Benutzerschnittstelle auf OSF/Motif unter UNIX/386*, Diplomarbeit Nr. 1157, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992.

[92] Kienle, M.: *Aufgespürt – Public-Domain-Software für die Überwachung und Analyse von Datenflußstörungen*, iX Multiuser-Multitasking Magazin, Nr. 9/94, September 1994, Seite 70-81.

[93] Kleinrock, L.: *Queueing Systems*, Vol. 1: Theory, John Wiley & Sons, New York, 1975.

[94] Klerer, M.: *The OSI Management Architecture: an Overview*, IEEE Network Magazine, Vol. 2, No. 2, März 1988, Seite 20-29.

[95] Kocher, H.: *Entwurf und Implementierung einer Simulationsbibliothek unter Anwendung objektorientierter Methoden*, 52. Bericht über verkehrstheoretische Arbeiten, Dissertation, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1994.

[96] Kreutzer, W.: *Grundkonzepte und Werkzeugsysteme objektorientierter Systementwicklung*, Wirtschaftsinformatik, Vol. 32, Nr. 3, Juni 1990, Seite 211-227.

[97] Kühn, P.: *Nachrichtenvermittlung II*, Vorlesung, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1989.

[98] Lederer, A.: *Aufbau einer Expertensystemshell für ein modulares verteiltes Diagnosesystem auf dem Betriebssystem UNIX/XENIX*, Diplomarbeit Nr. 1046, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990.

[99] Levine, J.; Mason, T.; Brown, D.: *lex & yacc*, Second Edition, O'Reilly & Associates, Sebastopol, CA, 1992.

[100] Lohmiller, R.: *Entwurf und Implementierung eines Monitoringsystems für ein verteiltes LAN-Meßsystem*, Diplomarbeit Nr. 1079, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1991.

[101] Manger, J.: *The World Wide Web, Mosaic and More*, McGraw-Hill, 1994.

[102] Maurer, H.: *Theoretische Grundlagen der Programmiersprachen: Theorie der Syntax*, BI-Hochschultaschenbücher, Mannheim 1969.

[103] McCloghrie, K.; Rose, M.: *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, RFC1213, März 1991.

[104] Moayeri, B.; Kell, H.; Suppan-Borowka, J.: *NQA Network Quality Analyzer*, Datacom, Vol. 8, Nr. 3, März 1989, Seite 22-28.

[105] Mogul, J.: *Efficient Use of Workstations for Passive Monitoring of Local Area Networks*, Proceedings Sigcomm '90 Symposium „Communications Architectures & Protocols", Philadelphia, September 1990, Computer Communications Review, Vol. 20, Nr. 4, ACM Press, 1990, Seite 253-263.

[106] Meshki, S.: *Protokolltester im Wandel*, Datacom, Vol. 11, Nr. 5, Mai 1994, Seite 26-28.

[107] Ousterhout, J.: *Tcl and the Tk Toolkit*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

[108] Orleth, R.: *Integration der Softwarekomponenten des verteilten modularen LAN-Meßsystems*, Diplomarbeit Nr. 1148, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992.

[109] Peuser, S.: *Netzmanagement in einem großen, heterogenen Netz*, Diplomarbeit Nr. 882, Institut für Parallele und Verteilte Höchstleistungsrechner/Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992.

[110] Postel, J.: *User Datagram Protocol*, RFC 768, August 1980.

[111] Postel, J.: *Internet Control Message Protocol*, RFC 792, September 1981.

[112] Protogeros, A., Ball, E.: *Traffic analyser and generator – Part 1: High-speed traffic capture for IEEE 802.3/Ethernet networks*, Computer Communications, Vol. 13, Nr. 7, September 1990, Seite 407-413.

[113] Protogeros, A., Ball, E.: *Traffic analyser and generator – Part 2: Traffic capture, generation, statistics and network integrity tests program in C*, Computer Communications, Vol. 13, Nr. 8, Oktober 1990, Seite 469-477.

[114] Rößler, G.; Schollenberger, W.: *Netzmanagement in heterogenen lokalen Netzen*, Praxis der Informationsverarbeitung und Kommunikation (PIK), Vol. 14, Nr. 4, Oktober 1991, Seite 211-215.

[115] Rodrigues, M.: *Erasure Node: Performance Improvements for the IEEE 802.6 MAN*, Proceedings IEEE INFOCOM '90, San Francisco, CA, Juni 1990, Seite 636-642.

[116] Romkey, J.: *Nonstandard for transmission of IP datagrams over serial lines: SLIP*, RFC 1055, Juni 1988.

[117] Rose, M.: *The Simple Book: An Introduction to Management of TCP/IP-based Internets*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[118] Rose, M.; McCloghrie, K.: *Structure and Identification of Management Information for TCP/IP-based Internets*, RFC 1155, Mai 1990.

[119] RzK: *RzK Net-Recorder*, Produktinformationen, RzK Doris Köpke, Ansbach.

[120] Sahin, V.: *Telecommunications Management Network: Principles, Models, and Applications*, in: A. Salah, T. Plevyak: Telecommunications Network Management into the 21st Century: Techniques, Standards Technologies, and Applications, IEEE Press, New York, 1994

[121] Santa Cruz Operation: *The SCO STREAMS System*, in SCO UNIX System V/386 Development System Programmers Guide, Santa Cruz Operation, 1989.

[122] Santa Cruz Operation: *The STREAMS Network Programmer's Guide*, in SCO UNIX System V/386 Development System Programmers Guide, Santa Cruz Operation, 1989.

[123] Schmid, M.: *Entwurf und Implementierung von Meßapplikationen für ein verteiltes LAN Meßsystem*, Diplomarbeit Nr. 1092, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1991.

[124] Schneider, J.: *Realisierung und Leistungsuntersuchung eines Kommunikationsmoduls zur Einbindung programmierbarer Steuerungen in den Informationsfluß eines Leitsystems unter Verwendung des MAP-Protokollstacks*, Diplomarbeit Nr. 1188, Institut für Nachrichtenvermittlung und Datenverarbeitung/ Institut für Steuerungstechnik und Werkzeugmaschinen, Universität Stuttgart, 1992.

[125] Schollenberger, W.: *Monitoringsystem für ein verteiltes LAN-Meßsystem*, ausgewählte Beiträge zum Workshop „Entwicklungstendenzen von Rechnernetzen", Fakultät Informatik der Technischen Universität Dresden, Gaußig 1991.

[126] Schollenberger, W.: *Netzmanagement*, in: Kommunikationstechnik für den rechnerintegrierten Fabrikbetrieb, P. Kühn, G. Pritschow (Hrsg.), Springer/TÜV Rheinland 1991.

[127] Schröder, J.: *Monitoring und Diagnose in lokalen Netzen*, Elektronik Nr. 7/91, Juli 1991, Seite 158-164.

[128] Schröder, J.: *Wissensbasierte Diagnose Lokaler Netze im Wirkbetrieb*, 54. Bericht über verkehrstheoretische Arbeiten, Dissertation, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1993.

[129] Seliger, P.: *Entwicklung einer grafischen Benutzeroberfläche für das Monitoringsystem unter OSF/Motif*, 2. Semesterarbeit Nr. 1324, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1994.

[130] Stallings, W.; Smith, B.: *SNMP Version 2*, Byte, Vol. 19, Nr. 8, August 1994, Seite 191-192.

[131] Staude, M.: *Planverfahren, Prozeßscheduling am Beispiel von Solaris 2.x*, iX Multiuser-Multitasking Magazin, Nr. 8/94, August 1994, Seite 130-136.

[132] Stine, R.: *FYI on a Network Management Tool Catalog: Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices*, RFC 1147, April 1990.

[133] Stuhler, F.: *Leistungsuntersuchung von Mehrprozeßsystemen unter UNIX*, 2. Semesterarbeit Nr 1191, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992

[134] *UNIX Programmer's Manual, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version*, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1983.

[135] Waldbusser, S.: *Remote Network Monitoring Management Information Base*, RFC 1271, November 1991.

[136] Wandel & Goltermann: *Lan, Wan & DA30 – Meßtechnik an Lokalen Netzen und Öffentlichen Netzen*, Seminarunterlagen, Wandel & Goltermann, Juli 1991.

[137] Weixler, M.: *Ein Konzept zur Leistungsmessung in verteilten Rechensystemen mit dezentralen Zeitgebern*, 53. Bericht über verkehrstheoretische Arbeiten, Dissertation, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1993.

[138] Yemini, Y.: *The OSI Network Management Model*, IEEE Communications Magazine, Vol. 31, Nr. 5, Mai 1993.

[139] Zborschil, B.: *Entwurf und Implementierung eines Befehlsinterpreters für eine Benutzer-Managementschnittstelle*, 2. Semesterarbeit Nr. 1286, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1994.

[140] Zieher, M.; Zitterbart, M.: *NETMON – a Distributed Monitoring System*, Proceedings 6th European Fibre Optic Communications & Local Area Networks Exposition (EFOC/LAN 88), Amsterdam, Netherlands, Juni/Juli 1988, Seite 452-457.

Anhang

Spezifikation der Meßauftragsbeschreibungssprache

```
/*************************************************************************
 *  Beschreibung der Zuordnung der definierten Zeichenfolgen zu den Tokens
 *
 *  Gewonnen aus dem Eingabefile fuer die Generierung des Scanners durch
 *  das Programm "lex".
 *
 *  (Syntax dieser Beschreibung nahezu selbsterklaerend)
*************************************************************************
 */

/* Parameterangaben */

TIME|[Tt]ime                          TIME
AFTER|[Aa]fter                        AFTER
AT|[Aa]t                              AT
TIMES|[Tt]imes                        TIMES
DURING|[Dd]uring                      DURING
UNTIL|[Uu]ntil                        UNITL
SUBTIME|[Ss]ubtime                    SUBTIME
RESULT|[Rr]esult                      RESULT
DISCARD|[Dd]iscard                    DISCARD
DURATION|[Dd]uration                  DURATION
INTERVAL|[Ii]nterval                  INTERVAL


SUM|[Ss]um                            SUM
MAX|[Mm]ax                            MAX
MIN|[Mm]in                            MIN
ALL|[Aa]ll                            ALL
STATISTICS|[Ss]tatistics              STATISTICS
MEAN|[Mm]ean                          MEAN

/* Stationsliste */

STAT|[Ss]tat                          STAT
\-\>                                  ARROW
GROUP|[Gg]roup                        GROUP

/* Bezeichungen der Messapplikationen */

REFL                                  REFL /* Reflektometer */
DF                                    DF    /* Durchflussmesser */
ECB                                   ECB   /* Echo-Box */
```

```
/* Komplexer Parameter Output */

OUTPUT|[Oo]utput                          OUTPUT
PID|[Pp]id                                PID
PROC|[Pp]roc                              PROC

/* Startzeitpunkt der EAB */

SHELL|[Ss]hell                            SHELL
WHEN_DONE|[Ww]hen_[Dd]one                 WHEN_DONE
AT_ONCE|[Aa]t_[Oo]nce                     AT_ONCE

/* Ablagemode der Zwischenergebnisse */

OVERWRITE|[Oo]verwrite                    OVERWRITE
SAVE_ALL|[Ss]ave_[Aa]ll                   SAVE_ALL
QUEUE|[Qq]ueue                            QUEUE
QUEUE_AND_SAVE|[Qq]ueue_[Aa]nd_[Ss]ave    QUEUE_AND_SAVE

/* Komplexer Parameter fuer die Spezifikation der
 * Messdurchfuehrung
 */

MEAS|[Mm]eas                              MEAS

/* Endekennung der Eingabe */

exit                                      EXIT
q                                         EXIT

/* Messauftragstypen */

/* Durchflussmesser */
df_count                    DF_COUNT      /* Pakete */
cable_count                 CABLE_COUNT   /* Aktivitaeten */
cable_test                  CABLE_TEST    /* Test des AUI-Anschl. */

refl                        MA_REFL       /* Reflektionsmessung */
echo                        MA_ECHO       /* Echotest ausloesen */
echo_new                    ECHO_NEW      /* Neuer Eintrag in
                                             Echotest-Tabelle */
echo_remove                 ECHO_REMOVE   /* Eintrag löschen */

report_config               REPORT_CONFIG /* Ausgabe der Konfiguration
                                             der Meßstation*/
connected                   CONNECTED     /* Prüfe ob Reflektometer
                                             vorhanden */
power                       POWER         /* Tap Stromzufuhr kontr. */
stop                        STOP          /* Messauftrag abbrechen */
jam_send                    JAM_SEND      /* JAM-Signal senden */

/* Messauftragsspezifische Parameter */

BACKGROUND|[Bb]ackground                  BACKGROUND
BACK_SEND|[Bb]ack_[Ss]end                 BACK_SEND
OFF_AT|[Oo]ff_[Aa]t                       OFF_AT
SMOOTH|[Ss]mooth                          SMOOTH
```

```
/* Definition der Zeiteinheiten  */

{S}\.{S}\.{S}[ ]*[,][ ]*{S}:{S}(:{S})?          Time      /* Datum/Uhrz. */
{S}:{S}(:{S})?                                 Time      /* nur Datum */
{I}[ ]?HRS([ ]?{I}[ ]?MIN([ ]?{I}[ ]?SEC)?)?   Duration  /* (lang) */
{I}[ ]?MIN([ ]?{I}[ ]?SEC([ ]?{I}[ ]?MS)?)?    Duration  /*  (kurz) */
{I}[ ]?SEC([ ]?{I}[ ]?MS)?                     Duration
{I}[ ]?MS                                      Duration


/* Stations-Adressen */

0x[ ]?(({B}[ ]?){5})({B})                      St_address
(({B}[ ]?{B}[ ]){2})({B}[ ]?{B})               St_address
{I}\.{I}\.{I}\.{I}\.{I}\.{I}                   St_address

/* Adresse der Messapplikation */
0x[ ]?({B})                                    Daap


/* variable Werte */
{I}                                            Integer
{N}                                            Name
\{[^\}]*\}                                     Op_list /* Parameter
                                                        * fuer die EAB */


/*************************************************************************
 *  Definition der Synax in einer BNF aehnlichen Syntax
 *
 *  Gewonnen aus dem Eingabefile fuer die Generierung des Parsers durch
 *  das Programm "yacc".
 *
 *************************************************************************
 */
/*  Bemerkung: Das Symbol „.“ (whitespace) erlaubt den Eingabestrom an
 *             vielen Stellen mit einen Carriage-Return-Zeichen zu
 *             strukturieren.
 */

more_ma:        EXIT                    /* Ende */
                | . ma '\n'             /* Meßauftrag */
                | error '\n'            /* Fehler */
                ;

ma:             df_count                /* s. o. */
                | cable_count
                | cable_test

                | refl

                | echo
                | echo_new
                | echo_remove

                | report_config
                | connected
                | power
                | jam_send

                | stop
                ;
```

```
/*------ Syntaxdiagramm: df_count ------------------------------------*/

df_count:       DF_COUNT .
                '(' . indep_parms df_parms ')'
                ;
df_parms:       BACKGROUND .
                | BACK_SEND .
                | OFF_AT . Integer . '%' . c_interval
                | /* nichts */
                ;

cable_count:    CABLE_COUNT . '(' . indep_parms ')'
                ;

cable_test:     CABLE_TEST . '(' . indep_parms ')'
                ;

refl:           MA_REFL . '(' . indep_parms SMOOTH . ')'
                | MA_REFL . '(' . indep_parms ')'
                ;

echo:           MA_ECHO . '(' . indep_parms ')'
                ;

echo_new:       ECHO_NEW . '(' . indep_parms ')'
                ;

echo_remove:    ECHO_REMOVE . '(' . indep_parms ')'
                ;

report_config:  REPORT_CONFIG . '(' . indep_parms ')'
                ;

connected:      CONNECTED . '(' . indep_parms ')'
                ;

power:            POWER . '(' . ON . indep_parms ')'
                | POWER . '(' . OFF . indep_parms ')'
                ;

stop:           STOP . '(' . JID . Integer . indep_parms ')'
                ;

jam_send:       JAM_SEND . '(' . indep_parms ')'
                ;

/*------ Syntaxdiagramm: indep_parms ---------------------------------*/

indep_parms:     indep_llist
                ;

indep_llist:    /* nichts */
                | indep_llist
                  one_ip
                ;

one_ip:         time .
                | stat .
                | output .
                | meas .
                ;
```

```
/*------ Syntaxdiagramm: time[]-----------------------------------------*/

time:           TIME . '[' . t_parms ']'
                ;

t_parms:        c_start  c_per_spec
                c_subtime  c_interval
                ;

c_start:        /* nichts */
                | start
                ;

c_per_spec:     /* nichts */
                | per_spec
                ;

c_subtime:      /* nichts */
                | subtime
                ;

c_interval:     /* nichts */
                | interval
                ;

/*------ Syntaxdiagramm: start -----------------------------------------*/

start:          AFTER  . Duration .
                | AT   . Time .
                ;

/*------ Syntaxdiagramm: per_spec --------------------------------------*/

per_spec:       period
                | period end
                | period count
                | end period
                | end count
                | count period
                | count end
                ;

/*------ Syntaxdiagramm: period ----------------------------------------*/

period:         ALL . Duration .
                ;

/*------ Syntaxdiagramm: count -----------------------------------------*/

count:          Integer . TIMES .
                | Integer . '*' .
                ;

/*------ Syntaxdiagramm: END -------------------------------------------*/

end:            DURING . Duration .
                | UNTIL . Time .
                ;
```

```
/*------ Syntaxdiagramm: subtime[] ----------------------------------*/

subtime:        SUBTIME . '[' . subper_spec
                c_result ']' .
                ;

subper_spec:    period count
                | period
                | count period
                | count
                ;

c_result:       /* nichts */
                | result
                ;

/*------ Syntaxdiagramm: result -------------------------------------*/

result:         RESULT . res_par        { $$=$3; }
                ;

res_par:        MAX .
                | MIN .
                | ALL .
                | STATISTICS .
                | MEAN  . c_discard
                ;

c_discard:      /* nichts */
                | DISCARD . Integer . '%' .

                ;

/*------ Syntaxdiagramm: interval -----------------------------------*/

interval:       DURATION  . Duration .
                | INTERVAL . Duration .
                ;

/*------ Syntaxdiagramm: stat[] -------------------------------------*/

stat:           STAT .
                '[' . list ']'
                ;

list:           listel el_sublist
                ;

el_sublist:     /* nichts */
                | el_sublist listel
                ;

listel:         st_c_list c_substat
                ;

st_c_list:      st_id
                | st_list
                ;

st_list:        '(' st_id
                 st_sublist ')'
                ;
```

```
st_sublist:     /* nichts */
                | st_sublist st_id
                ;

c_substat:      /* nichts */
                | ARROW . st_c_list
                ;

/*------ Syntaxdiagramm: st_id ----------------------------------------*/

st_id:          GROUP . Name . c_protlist
                | single_stat c_protlist
                ;

single_stat:    Name . st_subid
                | St_address . st_subid
                ;

st_subid:       /* nichts */
                | '-' . St_part .
                | '-' . Daap .
                ;

c_protlist:       /* nichts */
                | '<' . Name .
                  prot_sublist '>' .
                ;

prot_sublist:   /* nichts */
                | prot_sublist Name .
                ;

/*------ Syntaxdiagramm: output ---------------------------------------*/

output:         OUTPUT . '[' .
                   as_filename ']'
                | OUTPUT .
                  '[' . existent ']'
                | OUTPUT . '[' . shell ']'
                ;

as_filename:    Name . c_op_list
                   name_modes
                | Name . '@' . Name .
                  c_op_list  name_modes
                ;

name_modes:     WHEN_DONE .
                | c_at_once
                  c_modes
                ;

c_at_once:      /* nichts */
                | AT_ONCE .
                ;

c_modes:        /* nichts */
                | modes
                ;
```

```
existent:       PID .
                    Integer . c_modes
                | PROC . Name . c_modes
                ;

shell:          SHELL . c_s_modes
                ;

c_s_modes:      /* nichts */
                | s_modes
                ;

c_op_list:      /* nichts */
                | Op_list
                ;

/*------ Syntaxdiagramm: modes ----------------------------------------*/

modes:          s_modes
                | q_modes
                ;

s_modes:        OVERWRITE .
                | SAVE_ALL  .
                ;

q_modes:        QUEUE_AND_SAVE . c_msqid
                | QUEUE . c_msqid
                ;

c_msqid:        /* nichts */
                | ':' . Integer .
                ;

/*------ Syntaxdiagramm: meas -----------------------------------------*/

meas:           MEAS '[' Name '@' Name ']'
                | MEAS '[' Name ']'
                ;

/*------ Syntaxdiagramm: whitespace -----------------------------------*/

.:              /* nichts */
                | . '\n'
                ;
/*-------------------------------------------------------------------------*/
```