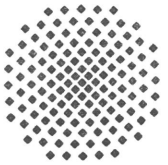


Universität Stuttgart

Institut für Nachrichtenvermittlung und Datenverarbeitung

Prof. Dr.-Ing. P. Kühn

53. Bericht über verkehrstheoretische Arbeiten

EIN KONZEPT ZUR LEISTUNGSMESSUNG IN VERTEILTEN RECHENSYSTEMEN MIT DEZENTRALEN ZEITGEBERN

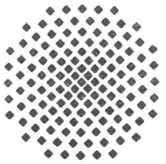
von

Michael Johannes Weixler

1993

Druck: E. Kurz & Co., Druckerei + Reprografie GmbH., Stuttgart

ISBN 3-922403-63-8



University of Stuttgart

Institute of Communications Switching and Data Technics

Prof. Dr.-Ing. P. Kühn

53th Report on Studies in Congestion Theory

A CONCEPT TO MEASURE PERFORMANCE IN DISTRIBUTED SYSTEMS USING DECENTRALIZED TIMERS

by

Michael Johannes Weixler

1993

A Concept to Measure Performance in Distributed Systems using Decentralized Timers

Summary

In today's local area networks, the bottleneck in performance is identified in processing the communication protocols of interconnected systems. Simply observing the common communication channel is not sufficient to gain insight into system internal dependencies and implementation specific delays.

A newly designed distributed measurement system based on spatial distributed hybrid monitor units allows measurements in real production environments. It is important to note, that no additional synchronization network or protocol is needed. By means of a prototype it is shown, that the influence on the systems under test is minimized and the accuracy is tuneable to the range of a few microseconds.

Chapter One

Typically used metrics to describe the performance of standalone computer systems are shown. Some metrics can be extended to measure distributed systems. To minimize the unknown influence of network load and network access delays, most approaches which are reported require empty networks and synthetic work loads (see table 1).

The new concept, described in the following chapters, enables the execution of concurrent measurements in several spatial distributed systems located in real environments.

Chapter Two

This chapter provides definitions for the terms *performance*, *event* and *measurement point*. The performance models of CCITT (see fig. 2.2) or OSI use these terms to specify derived attributes as described in section 2.2. Both performance models are investigated with regard to their applicability for performance measurements in all parts and levels of any system.

The preferred alternative concept is based on *monitors*. Integrating a hardware monitor and software monitor into a hybrid monitor yields the means to trace all actions in the system under test.

Chapter Three

Chapter three introduces some problems that arise, when a hybrid monitor is extended to a distributed monitor system.

Measurements in distributed systems require a network wide view, thus requiring a global time basis. Fig. 3.3 presents a classification scheme for time synchronization mechanisms. For each of these synchronization mechanisms different ways exist to build a measurement system. The pros and cons of each approach are briefly discussed.

Furtheron, the problem of synchronously starting a distributed measurement task is addressed.

Analyzing collected measurement data depends on the way a measurement system encodes and stores the data. The two possible solutions of centralized versus decentralized storage are compared. The storage solution also determines the way, a user may analyze the measurement data, either online or offline.

Because a distributed measurement may collect huge amounts of data, dedicated tools to support the user in analyzing the data are needed, as stated in sections 3.5.

The same applies for running a measurement task with a distributed measurement system.

Chapter Four

This chapter describes in great detail the characteristics of this new design. The reasons for choosing a hybrid monitor, selected measurement points or the way to pass measurement data via *monitor registers* to a measurement unit are explained.

Fig. 4.5 shows a measurement unit with its local timer used to time stamp each collected data item. Synchronization of all timers is done after a measurement task using special entries in the *event traces*.The algorithm to resynchronize time values is explained in section 4.3. All steps of this algorithm are shown in a graphical representation in fig. 4.12. This resynchronization is needed to build a global event trace of all collected events, sorted by increasing global time.

The global event trace is the input to a parser, which recognizes events described in a powerful data description language (DDL). The DDL permits to group events in pairs or ordered sequences for further analysis, using a statistical interpreter.

Chapter Five

A prototype was built to show the realization with low costs.

Some hardware sensors are explained in greater detail: specifically a sensor to trace the actions of a media access controller and a very high integrated chip version of a sensor (fig. 5.8) used to recognize patterns on the common channel. The latter sensor supports the resynchronization algorithm, described in chapter four.

A second type of sensors, a monitor register, was developed for a few standard bus systems. The various versions of the sensor and the access statements are compared in following sections.

Section 5.4 explains the communication prtotocol used to transfer measurement data and control messages between measurement unit and a central analyzing station. The protocol is similar in function to ISO/OSI LLC type 3, but allows acknowledged multicast transfers. This feature is used to spread measurement start messages.

The remaining part of chapter 5 shows the steps from time resynchronization to time interpolation and sorting of events, done by several fault tolerant tools running on the central analyzing station. The description of parser implementation and statistical interpreter closes this chapter.

Chapter Six

Accuracy and resolution are derived for the prototype implementation of chapter five. This includes all delays using either hardware sensors or software statements plus monitor registers. Table 18 lists the runtime of typical software statements to access a monitor register.

The influence of available quartz oscillators on the achievable accuracy is analysed and the possible loss of event entries used for resynchronization are taken into account.

Finally, observed trace lengths and processing times to analyze the data are listed on page 133.

Chapter Seven

This chapter shows the usage of the distributed measurement system for some selected examples.

Section 7.1 deals with an operating system extension, known as the network interface driver. More complex examples including a management gateway (fig. 7.2) coded as a multitasking application and an application including all protocol layers down to layer 2 (fig. 7.6) are presented in the following subsections.

A simple file transfer is used in section 7.2 to show the problems, which arise using this measurement system in distributed applications.

Chapter Eight

The concept has shown, that measurements are possible in all layers and components of distributed systems.

The accuracy is sufficient and is of the same order of an access to a monitor register.

The outlook lists some possible improvements to increase the time accuracy at the cost of more complex and expensive sensors and measurement units.

By nature, all measurements in the real environment cannot be reproduced, even if tools like a distributed load generator are used. Therefore, the distributed measurement system and all its related tools try to support the user in analyzing his unique data to get more insight into system internal behaviour.

Inhaltsverzeichnis

Abkürzungen

ACSE	Association Control Service Element
AUI	Attachment Unit Interface
BDTP	Bulk Data Transport Protocol
BNF	Backus-Naur Form
CAD	Computer-Aided Design
CAE	Computer Aided Engineering
CAM	Computer-Aided Manufacturing
CAP	Computer-Aided Production Planning
CAQ	Computer-Aided Quality Control
CASE	Common Application Service Element
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CIM	Computer-Integrated Manufacturing
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Entity
CNMA	Communications Network for Manufacturing Applications
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense, Multiple Access with Collision Detection
DDL	Data Description Language
DMA	Direct Memory Access
DSAP	Destination Service Access Point
DSE	Data Switching Equipment
DTE	Data Terminal Equipment
EPA	Enhanced Performance Architecture
FTAM	File Transfer, Access and Management
FTP	File Transfer Protocol
GDT	Global Descriptor Table
IHI	Independent Host Interface
iRMX	Intel Realtime Multitasking Executive
IP	Internet Protocol
ISO	International Organization for Standardization
ISO 2a	ISO/OSI - Medienzugangsschicht
ISO 2b	ISO/OSI - logische Sicherungsschicht
ISO 3	ISO/OSI - Vermittlungsschicht
ISO 4	ISO/OSI - Transportschicht
ISO TP4	ISO/OSI - Transportschicht, Klasse 4
LAN	Local Area Network
LFSR	Linear Feedback Shift Register
LLC	Logical Link Control
LSAP	Link Layer Service Access Point

MAC	Media Access Control
MAP	Manufacturing Automation Protocol
MIB	Management Information Base
MIP	MULTIBUS Interprocessor Protocol
MIPS	Million Instructions per Second
MMS	Manufacturing Message Specification
MMFS	Manufacturing Message Format Standard
OSI	Open System Interconnection
PAL	Programmable Array Logic
PDU	Protocol Data Unit
RPC	Remote Procedure Call
RSR	Rückgekoppeltes Schieberegister
SCB	Status Control Block
SDL	Specification and Description Language
SNMP	Simple Network Management Protocol
SSAP	Source Service Access Point
SUT	System Under Test
TCP	Transmission Control Protocol
TSAP	Transport Service Access Point
TPDU	Transport Protocol Data Unit
TTL	Transistor-Transistor Logik
UDP	User Datagram Protocol
VLSI	Very Large Scale Integration
VMTP	Versatile Message Transaction Protocol

Formelzeichen

Δt_i	Interner zeitlicher Abstand in einer Musterserie
ΔT	Zeitlicher Abstand der begrenzenden Muster einer Musterserie oder eines Musterpaares
ΔZ	Maximale Uhrenabweichung
$\lfloor Z \rfloor$	Größte ganze Zahl kleiner oder gleich Z
c	Korrelationskoeffizient
d	Fehlerabweichung
f	Teilerfaktor für Frequenz
g	Globale Meßfühleradresse
l	Lokale Meßfühleradresse
s	Anzahl wartender Ereignissse am Abfragebus
$A \rightarrow B$	Paarzuordnung: Muster A gefolgt von Muster B
D	Dynamisches Korrelationsintervall
F	Oszillatorfrequenz
F^i	Folge der SYNC-Zeitintervalle aus Spur i
F_{Ein}	Einschreibfrequenz
F_{Aus}	Auslesefrequenz

ID	Meßfühleridentifikation
K	Anzahl von Korrelationsrechnungen
K_i	Obere Schanke der Klasse i
$K_{Schritt}$	Klassenschrittbreite
L	Länge des Meßspeichers
L_{real}	Reale FIFO-Speicherlänge
L_{virt}	Virtuelle FIFO-Speicherlänge
M	Meßdauer
M[i,j]	Wert aus Verschiebematrix von Spur i mit Spur j
R_1	Maximale Meßdatenrate je Meßfühler
R_Z	Resynchronisationsrate i.A. von ΔZ
S	Statisches Korrelationsintervall
T	Zählerzyklusdauer
T_A	Auftragsbearbeitungszeit
T_B	Geschätzte Bearbeitungszeit
T_E	Erfassungszeit
T_{Eu}	Untere Grenze der Erfassungszeit
T_{Eo}	Obere Grenze der Erfassungszeit
T_H	Hinlaufzeit
T_i	Bearbeitungszeit
$T_{Abstand}$	Minimaler Rahmenabstand
T_{Bus}	Verzögerung im Systembus-Meßfühler
T_D	FIFO-Durchfallzeit
T_{K6}	Verzögerungszeit bei Transfererkennung
T_{K7}	Verzögerungszeit bei SYNC-Erkennung
T_M	Abfragezeit je Meßfühler
T_R	Rücklaufzeit
T_{Rahmen}	Minimale Rahmendauer
T_{RE}	Verzögerung im Meßfühler bei SYNC-Erkennung bis Rahmenende
T_{SCB}	Zeit bis Zugriff auf Status Control Block (SCB)
T_{SR}	Verzögerung im Meßfühler bei Transfererkennung
T_{SYNC}	Maximal erlaubter SYNC-Abstand
T_v	Rahmenverzögerung bei kollisionsbehaftetem Senden
T_j^i	j-tes Ereignis in Spur i oder j-tes Abstandsintervall zwischen SYNC-Ereignissen in Spur i
Z	Länge des Abfragezyklus
ZS	Zeitstempelinhalt

1 Einleitung

Mit dem zunehmenden Einsatz von Rechnern im täglichen Leben werden immer mehr Anforderungen an diese Alleskönner gestellt. So finden heute Rechner neben den klassischen Einsatzbereichen wie der numerischen Mathematik weiter große Verbreitung in Bereichen vom computerunterstützten Entwurf (engl. Computer-Aided Design, CAD) über die computerunterstützte Arbeitsvorbereitung (engl. Computer-Aided Production Planning, CAP) bis hin zur computerunterstützten Fertigung (engl. Computer-Aided Manufacturing, CAM) und Qualitätskontrolle (engl. Computer-Aided Quality Control, CAQ), die alle unter dem Oberbegriff computerunterstützte Ingenieurtätigkeit (engl. Computer-Aided Engineering, CAE) zusammengefaßt werden. Wird eine gemeinsame Datenbasis auch für betriebswirtschaftliche Aufgaben genutzt, so spricht man von computerintegrierter Fertigung (engl. Computer-Integrated Manufacturing, CIM) [40]. Gerade beim Einsatz im CIM-Bereich kommt es darauf an, daß alle Komponenten, seien es die eingesetzten Rechner (Hardware-Basis), die eingesetzten Anwendungsprogramme (Applikations-Software) oder die zur Vernetzung nötigen Komponenten wirtschaftlich und aufeinander angepaßt sind. Neben der Verfügbarkeit standardisierter oder branchenspezifischer Anwendungsprogramme sind es zunehmend Ergonomie- und Leistungsanforderungen, die entscheidend sind bei der Auswahl von Rechenanlagen.

Soll eine Rechenanlage von nicht rechnerspezifisch ausgebildetem Personal bedient werden, so muß die Bedienung dieses Systems benutzerfreundlich ausgelegt sein. Wird darüber hinaus die einzelne Rechenanlage im Verbund mit weiteren Rechenanlagen betrieben, so sollte diese Vernetzung für den Benutzer nicht sichtbar sein. Ergonomie verlangt somit zusätzliche Rechenleistung zur Bereitstellung einer benutzerfreundlichen, möglichst grafischen, Mensch-Maschine-Schnittstelle.

Leistungssteigernde Maßnahmen sind der Einsatz besserer Technologien, effizientere Systemarchitekturen und neue Programmiertechniken. Durch den Einsatz von neuen höchstintegrierten Bausteinen beim Schaltungsentwurf, der rechnerunterstützen Umsetzung in fertige Baugruppen durch CAE-Werkzeuge und der objektorientierten Programmiersprachen zur Softwareerstellung kann ein Entwickler die heute geforderten kurzen Entwicklungszeiten einhalten, die im Konkurrenzkampf mit anderen Herstellern nötig sind.

In den letzten Jahren hat sich die Leistung der Mikroprozessoren alle 5 Jahre verzehnfacht, mit Multiprozessor- und vernetzten Mehrrechnersystemen, die gemeinsam an der Berechnung einer Aufgabe arbeiten, wird die Leistungsfähigkeit nochmals gewaltig gesteigert [44]. Dabei verlagert sich der heutige Forschungsschwerpunkt im Bereich der leistungsstärksten Rechenanlagen von den Monoprozessorsystemen hin zu parallel arbeitenden Multiprozessor- und Mehrrechnersystemen, wobei die Kommunikationsfragen einen weiten Raum einnehmen.

Neben den vielen proprietären Netzwerkarchitekturen [38, 150, 175, 26, 210], die sich im Laufe der Jahre, ausgehend von einfachen Sicherungsverfahren bis zu neuesten Transaktionsprotokollen und dezentralen virtuellen Dateisystemen, entwickelt haben, gibt es eine Reihe von standardisierten Protokollen, die für unterschiedlichste Netzwerke entworfen wurden.

1.1 Kommunikationsarchitekturen

Beim Systementwurf stellt sich somit die Frage, welche Kommunikationsprotokolle zueinander passen und, aufeinandergeschichtet, die Protokollsäule (engl. Protocol Stack) bilden sollen. Sinnvolle Auswahlen aus der Vielzahl von Kombinationsmöglichkeiten wurden von Firmengremien (unter Leitung von General Motors als Manufacturing Automation Protocol (MAP) [129] für den CIM-Bereich oder von Boeing als Technical and Office Protocols (TOP) [171] für Büroberei-che) Anfang der achtziger Jahre vorgeschlagen und allgemein übernommen [74], jedoch immer wieder in Frage gestellt [191].

Außer dem von ISO im OSI-Basisreferenzmodell [96] beschriebenen 7-schichtigen Protokollstack gibt es standardisierte Alternativen von CCITT [19], der Internet-Community [168] oder dem DIN (Profibus) [112, 162], die eine abweichende Schichteneinteilung haben.

Beim Verbund von Rechenanlagen gibt es nach [193] vier Klassen der Rechnervernetzung in Richtung erhöhter Integration, basierend auf:

1. Rechnern, die an ein zentrales System sternförmig angeschlossen sind;
2. autonomen Rechnern mit Netzzugang und einem Kommunikationsprotokoll [57, 165];
3. autonomen Rechnern nach Klasse 2 mit einem integrierten Dateisystem [175];
4. Rechnern mit verteilten Betriebssystemkernen.

Alle Systeme der interessanten Klassen 2-4 besitzen neben den lokalen Betriebsmitteln Zugang zu einem Netz, das oftmals den gleichzeitigen und parallelen Einsatz einer Vielzahl von Kommunikationsprotokollen erlaubt. Damit sind Netzübergangsstationen (engl. Gateways), die zwischen unterschiedlichen Netzen oder zur Anpassung unterschiedlicher Kommunikationsprotokolle erforderlich sind, in die Leistungsbetrachtung miteinzubeziehen.

Neben den reinen Kommunikationsleistungen eines Systems sind auch die Anforderungen heutiger [119, 144, 161] und zukünftiger Applikationen an die Systeme und das Netz von Interesse. Durch Analyse heutiger Anwendungen können Fragen der Kapazitätsplanung beantwortet [156], Systemkonfigurationen verbessert [123] aber auch der Leistungsbedarf zukünftiger Applikationen abgeschätzt werden [126, 207].

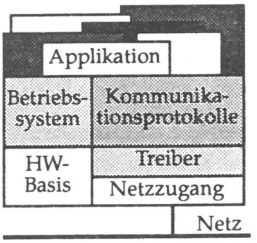


Bild 1.1: Komponenten eines zu beobachtenden Systems für die Messung

Zur vollständigen Angabe der Systemleistung ist notwendig, daß nicht nur Anwendungen oder nur das Betriebssystem oder nur Teile der Hardware-Basis leistungsmäßig untersucht werden, sondern alle auf einer Rechenanlage ablaufenden Vorgänge gleichzeitig. Darüberhinaus wird gezielt der Einfluß und die Leistung der zur Kommunikation notwendigen Systemerweiterungen wie Netzzugangsbaugruppen und der dazu nötigen Treiber und Kommunikationsprotokolle in Form von Hardware- oder Software-Erweiterungen untersucht (vergl. Bild 1.1). Dies setzt voraus, daß die zu untersuchenden Systeme (engl. System Under Test, SUT) in einem realen Netzwerk operabel sind und aufgrund realer oder synthetischer Applikationen Daten über das Netz austauschen.

Werden, wie im CIM-Bereich, viele Rechner vernetzt, so sind Leistungsangaben über diesen Verbund nötig, die die Kommunikationsleistungen des Gesamtsystems miteinbeziehen. Wichtige Angaben bezüglich des Netzes, welche die Leistung beeinflussen, sind:

- Eigenschaften des Netzes, wie z.B.
  Netztopologie,
  Transferrate,
  Bitfehlerrate,
  Signallaufzeiten,
  Datentransferbetriebsmodus (uni-, bidirektional),
- Protokolleigenschaften, wie z.B.
  Paketgrößen,
  Fenstergrößen,
  Kanalzugriffsverfahren,
  Nachrichtenprioritäten,
  Fairness,
- Nutzungsmerkmale, wie z.B.
  Dienste,
  Anwendungen,
  Tageszeit,
  Verkehrsaufkommen.

Die Leistung des Netzes wird ausgedrückt in Metriken wie:

- Durchsatz,
- Auslastung,
- Wartezeiten oder
- Transferzeiten.

Bei den verwendeten Kommunikationsprotokollen interessieren ebenfalls Durchsatz und Wartezeiten, aber daneben auch die Betriebsmittelauslastung wie Kodeauslastung oder Speichereffizienz.

1.2 Ziel der Leistungsmessung

Jede Messung hat zum Ziel, die Leistung des betrachteten Systems möglichst genau zu erfassen und die Systembeeinflussung minimal zu halten. Durch Messungen mit wechselnder Belastung,

unterschiedlichen Systemeinstellungen und äußeren Einflüssen läßt sich die Reaktion der Rechenanlage genau bestimmen. Wird eine Rechenanlage im Fabrikbereich eingesetzt, so sind andere Belastungsfälle üblich als im Bürobereich, da die eingesetzten Applikationen und auch das Benutzerverhalten gänzlich andere sind [60, 61]. Deshalb wurden verschiedentlich Lastmodelle (engl. Workload Models) [48] definiert:

- Modelle in Abhängigkeit der Rechnerkopplung [49],
- Modelle für Filesystemoperationen [10],
- Modelle zur Bewertung von technologischen Abhängigkeiten [69] oder
- Modelle aus dem Fertigungsbereich [71].

Trotzdem möchte man vergleichbare Resultate erhalten, da eine Rechenanlage sowohl im Fabrikbereich als auch im Bürobereich einsetzbar ist. Dies ist durch vergleichbare Belastungen mittels parametrisierbarer Lastmodelle erreichbar, wenn auch die Messung nach einer einheitlichen Vorschrift [vergl. 59] durchgeführt wird.

Die gemessenen Resultatwerte sind für viele Kreise interessant:

• Marketing	Klassifizierung ähnlicher Rechenanlagen anhand einfacher Leistungswerte;
• Entwickler	Optimierung des Systems entsprechend den geforderten Systemleistungen;
• Systemplaner	Optimierung bestehender Anlagen durch Modifikationen der Systemarchitektur oder Verifizierung der analytisch bzw. simulativ gewonnenen Leistungswerte;
• Betreiber	Konfiguration von Rechenanlagen (auch im Verbund) oder Kapazitätsplanung aufgrund einfacher Regeln.

Die gemessenen Werte hängen im Falle einer untersuchten Rechenanlage von der aktuellen Konfiguration ab, d.h. neben dem hardwaremäßigen Ausbau der Anlage haben auch die installierten Software-Versionen und die Systemeinstellungen Einfluß auf die Leistungswerte.

Wird eine Messung im Rechnerverbund durchgeführt, so sind die Einflüsse der beteiligten anderen Anlagen zu bewerten und anzugeben. Diese externen Einflüsse können nicht so einfach quantitativ angegeben werden, da im Rechnerverbund alle Anlagen durch Wechselwirkungen miteinander in Beziehung stehen. Dies hat dazu geführt, daß viele Leistungsmessungen nur in bestimmter Umgebung, d.h. in Netzen mit einer einzelnen realen Anwendung, an ausgewählten Netzkomponenten oder mit gänzlich synthetisch erzeugtem Verkehr durchgeführt wurden (siehe Tabelle 1).

	Protokolle (Schnittstelle)	Stations-anzahl	Netz	Verkehr	Literatur
Netzwerk-Schnittstelle	VMTP	» 2	real	real	[28]
	ISO, TCP/IP, DNA	» 2	real	real	[60]
	TCP/IP	» 2	real	real	[67]
	DECwindows	» 2	real	real	[144]
	TCP/IP	» 2	real	real	[157]
Anwender-Schnittstelle	TCP	» 2	real	real	[8]
	ISO 2a	2	leer	synthetisch	[45]
	FTAM	2	leer	real	[27]
	VMTP	2	real	real	[29]
	ISO 4	2		synthetisch	[41]
	FTAM, FTP, RPC	2	real	real	[65]
	DECnet	2	leer/real	synthetisch	[69]
	ISO 4	» 2	leer	synthetisch	[70]
	MMS	2	leer	synthetisch	[124]
	ACSE	2	leer	synthetisch	[131]
	CASE, MMFS	2	leer	synthetisch	[146]
	ISO 2b, ISO 4	2	real	synthetisch	[158]
	IP, Mach RPC,	2	leer	synthetisch	[164]
	NFS	2	leer	real	[189]
	ISO 2b, ISO 4	2		synthetisch	[192]
	BDTP	2		real	[16]
System intern	TCP/IP	1	leer	real	[11]
	TCP/IP	1		real	[35]
	TCP	1	real	real	[32]
	Raidcomm	1		real	[128]
	SINEC AP	1	leer/real	real	[136]
	PUP, VMTP	1		real	[145]
	ISO 3	» 2		synthetisch	[42]

Tabelle 1: Leistungsmessungen an Kommunikationsprotokollen

Die Resultate einer durchgeführten Messung zeigt man im einfachsten Fall über Anzeigeelemente, etwa Leucht- oder Balkenanzeigen, direkt an. Weitere Formen sind Auflistungen in Tabellenform, als diskrete Verteilungen, als Gantt-Diagramm (Bild 1.2a) oder Häufigkeitsdiagramm einzelner Programmabschnitte P_i wie in Bild 1.2b und Bild 1.2c.

Um nur selten auftretendes Verhalten zu erfassen, sind lange Meßphasen nötig. Dies führt zu sehr vielen Rohdaten, die man üblicherweise mit Hilfe statistischer Methoden verdichtet. Aus der gemessenen Stichprobe berechnet man bei der deskriptiven Statistik die Lagegrößen:

- Extremwerte (Minimal- und Maximalwert),
- arithmetischer Mittelwert,

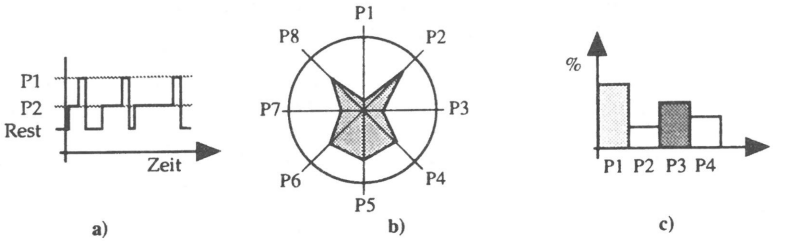


Bild 1.2: Ergebnisdiagramme: a) Gantt-Diagramm, b) Kiviat-Graph, c) Auslastungsprofil der ausgeführten Programmteile P_i

- Median (mittlerer Meßwert oder arithm. Mittel der beiden mittleren Klassenmeßwerte),
- n%-Quantil (Wert, unterhalb dessen n% aller gemessenen Werte liegen) und,
- Modalwert (Meßwert mit der größten Häufigkeit)

sowie die Streugrößen:

- Varianz und
- Standardabweichung.

Weiterhin lassen sich die Meßwerte in Klassen einteilen und in grafischer Form als Histogramme (vergl. Bild 1.2c) darstellen.

Mit diesen statistischen Größen kann das gemessene Objektsystem beschrieben werden, es können aber diese Resultate auch als Eingabewerte für Analysen und Simulationen eines Objektmodells dienen.

In Tabelle 1 sind einige veröffentlichte Arbeiten zum Thema Leistungsmessung in Kommunikationsprotokollen erfaßt, die entweder nur an einer Schnittstelle Leistungswerte messen konnten oder durch Eintragen des Eintreffzeitpunktes bei vordefinierten Meßpunkten in reservierte Felder innerhalb des zu transferierenden Datenpaketes selbst Messungen erlaubten. Beide Meßkonzepte sind nicht für den Einsatz in einer realen Umgebung geeignet, weil dort zur Leistungsmessung an einzelnen Komponenten eine netzweite Sicht erforderlich ist und die Änderung der Paketinhalte nicht akzeptabel ist. Weitere veröffentlichte Meßkonzepte, die Einsicht in systeminterne Vorgänge erlauben, stützten sich auf das Monitorkonzept, wobei verteilte Monitorkomponenten über ein eigenes Monitornetz sowie ein separates Synchronisationsnetz verbunden waren. Im Rahmen dieser Arbeit wird ein Konzept vorgestellt, das ebenfalls auf dem Monitorkonzept basiert, jedoch kein zusätzliches Netz verlangt und somit in einer realen Umgebung einsetzbar ist.

1.3 Übersicht über die Arbeit

In einem verteilten Rechensystem, bestehend aus einzelnen Rechenanlagen mit deren wesentlichen Komponenten Hardware-Basis, Betriebssystem, Kommunikationssystem und den Applikationen, sollen Leistungsmessungen an sämtlichen Komponenten aller beobachteten Rechenanlagen gleich-

zeitig möglich sein. Dabei gilt immer die Forderung, daß mit minimalem Aufwand sehr genaue Messungen innerhalb der Systeme möglich sein sollen.

In Kapitel 2 werden nach Einführung von grundlegenden Begriffen und Definitionen verschiedene Ansätze zur Leistungsmessung vorgestellt. Dazu werden die standardisierten Vorschläge, das von der ISO definierte OSI-Leistungsmeßkonzept und das im Internet angewandte Leistungsmeßkonzept, auf ihre Verwendbarkeit für die hier gestellten Anforderungen untersucht. Darüberhinaus wird das Monitorkonzept[1] in seinen verschiedenen Varianten den standardisierten Konzepten gegenübergestellt.

Kapitel 3 führt in die Meßproblematik in verteilten Systemen ein, beginnend bei Messungen in nur einem vernetzten System. Gleichzeitige Leistungsmessungen in mehreren Systemen verlangen, daß die Meßinformationen mit den Erfassungszeitpunkten markiert werden. Dies verlangt eine Zeitsynchronisation in den verteilten Systemen. Unterschiedliche Verfahren zur Synchronisation werden vorgestellt und miteinander verglichen. Die Synchronisation der Uhren hängt mit dem Speicherkonzept zusammen und daraus wiederum ergeben sich Anforderungen für den Transfer und die Darstellung der Information. Diese Darstellung ist möglicherweise aufgrund der hardwaremäßigen Erkennung oder den Benutzerwünschen kodiert, weshalb eine Informationsbeschreibung vorliegen muß. Bei bekannter Datendarstellung kann dann eine Auswertung der Meßinformationen stattfinden, wobei die Zuordnung von Benutzersicht zu gemessenen Werten entweder über eine geeignete Informationskodierung und Intelligenz in den Auswertungsprogrammen erfolgt oder dem Meßsystemanwender überlassen wird.

Kapitel 4 stellt die gewählte Architektur für ein neues verteiltes Meßsystem vor [205]. Ausgehend vom Hybridmonitor-Konzept werden die möglichen Meßfühler zum Informationsgewinn vorgestellt. Als dezentrale Monitorkomponente wird jedem Objektsystem ein intelligenter Sensor als Meßteil zugeordnet, der für Informationserfassung, Filterung, Speicherung und Vorverarbeitung zuständig ist. Als steuernder Teil dient eine zentrale Rechenanlage, die alle dezentralen Monitorkomponenten über ein spezielles Kommunikationsprotokoll steuert. Die Zeitsynchronisation, basierend auf freilaufenden Uhren und Resynchronisation anhand von Resynchronisationspunkten, wird erläutert, bevor abschließend die zur Informationsdekodierung entwickelte Beschreibungssprache und die damit verknüpfte Statistikauswertung beschrieben wird.

Kapitel 5 gibt eine Übersicht über die prototypische Realisierung der entwickelten Hardware-Baugruppen zur Informationserfassung und -speicherung. Anschließend wird auf die Meßsteuerung im verteilten Fall eingegangen, die aus zentralem Kontrollprogramm, Kommunikationsprotokoll und den dezentralen Steuerungen besteht. Ein Vorverarbeitungsschritt synchronisiert die verteilt anfallenden Ereignisspuren und faßt diese zu einer Ereignisspur zusammen, bevor ein Programm zur Semantikprüfung (engl. Parser) die Informationen in dieser globalen Ereignisspur dekodiert und in einer Referenzdatenbank ablegt. Schließlich liefert der zur Auswertung entworfene Statistikinterpreter die gewünschten Resultate.

---

1 Der Begriff Monitor wird anderweitig verwendet als Synonym für Betriebssystem oder für den Entscheidungsprozess beim Zugriff auf gemeinsam genutzte Daten (üblicherweise zur Steuerung der Zugriffssynchronisation) [81]. Hier wird unter einem Monitor ein Prozeß zur Extrahierung von sich ändernder Information aus einem laufenden System verstanden.

Die Bewertung der Güte und Möglichkeiten des verteilten Meßsystems folgt in Kapitel 6. Dazu wird die Meßgenauigkeit in Abhängigkeit des Synchronisationsverfahrens angegeben. Abstände und Häufigkeiten von in der Ereignisspur vorkommenden Resynchronisationspunkten beeinflussen die Resynchronisation, basierend auf Korrelationsverfahren, bezüglich eindeutiger Zuordnung der sich entsprechenden Resynchronisationspunkten. Simulative Ergebnisse der Nachsynchronisation, auch unter der Annahme verlorener Resynchronisationspunkte oder verschobener Startzeitpunkte, werden präsentiert. Eine wichtige Rolle für die statistische Auswertung spielt die Anzahl der Ereignisse, die auch entscheidend für die notwendige Berechnungsdauer beim Zusammenfassen der einzelnen Ereignisspuren zu einer globalen Ereignisspur und der Dateninterpretation ist.

Kapitel 7 zeigt Beispiele für den Einsatz des verteilten Meßsystems für lokale Messungen an einem Objektsystem und auch im verteilten System. Für die durchgeführten Messungen wurden unterschiedliche Anwendungen oder Teile des Betriebs- und Kommunikationssystems selbst ausgewählt, um die Anwendbarkeit, aber auch die dabei auftretenden Schwierigkeiten, auf den diversen Objektsystemen zu demonstrieren.

Das Schlußkapitel faßt die wesentlichsten Ergebnisse dieser Arbeit zusammen und gibt Ausblicke auf denkbare Erweiterungen.

2 Leistungsmessung

Bevor unterschiedliche Ansätze zur Leistungsmessung in vernetzten Systemen untersucht werden, muß durch Definitionen eine Basis für vergleichbare Leistungsmaße bereitgestellt werden.

2.1 Ansätze zur Leistungsmessung

2.1.1 Definitionen

DIN [37] definiert *Leistung* (engl. Performance) als *DV-Arbeit* pro Zeit, die ein *Auftraggeber* über eine *Schnittstelle* von einem *Auftragnehmer* ausführen läßt. Nach Ablauf der *Auftragsvorbereitung* wird ein *Auftrag* an der Schnittstelle zum Auftragssnehmer übergeben und dort für die Dauer der *Auftragsdurchlaufzeit* bearbeitet, bis das Endergebnis an derselben Schnittstelle bereitliegt. Mit jedem Auftrag sind Durchlaufzeitforderungen verknüpft wie vorgegebene, tolerierbare und nicht zu überschreitende Abarbeitungszeit. Damit ergibt sich als *erbrachte Leistung* der Quotient von zeitgerecht erbrachter Leistung je Meßdauer. Als Aufträge während einer Meßdauer werden alle innerhalb dieser Zeit gestarteten Aufträge gezählt, wobei ein eingeschwungener Systemzustand verlangt wird.

In [40] wird der Begriff Leistung von Rechensystemen beschrieben als:

„Geschwindigkeit und Qualität, mit der ein Auftrag oder eine Menge von Aufträgen von einer Datenverarbeitungsanlage verarbeitet wird. Zur Bewertung zieht man geeignete Meßgrößen heran, von denen die wichtigsten sind:

- Durchsatz: Zahl der pro Zeiteinheit bearbeiteten Aufträge;
- Antwortzeit: Zeit, die ein Auftrag vom Zeitpunkt seines Eintreffens bis zum Abgang nach erfolgter Bearbeitung im Datenverarbeitungssystem verbringt;
- Verfügbarkeit: Wahrscheinlichkeit, eine Anlage zu einem gegebenen Zeitpunkt in einem funktionsfähigen Zustand anzutreffen.

Zum Vergleich der Leistung unterschiedlicher Rechenanlagen objektiviert man die Messung obiger Größen durch die Verwendung von Bewertungsprogrammen, die unter einheitlichen Bedingungen auf verschiedenen Rechnern ausgeführt werden (vergl. KOp/s)."

Zum Vergleich unterschiedlicher Rechensysteme legt man entweder ein theoretisches Referenzsystem fest oder bezieht die Leistungsmaße auf ein reales Referenzsystem. Eine übliche Maßzahl, die obiger Beschreibung für den Durchsatz genügt, ist MIPS (engl. Million Instructions Per Second). Daß auch dieses einheitliche Maß nicht immer vergleichbare Werte liefert, zeigt [203], und in [125] werden die „sieben Fehler" beim Leistungsvergleich von Rechenanlagen aufgrund von *Bewertungsprogrammen* (engl. Benchmarks) angegeben. In [51] wird der Einfluß der Berechnungsart (arithmetisch oder geometrisch) auf den Mittelwert beschrieben, der meist zum Leistungsvergleich von Rechenanlagen dient. [179] zeigt Beispiele von Fehlern, Fallen oder Möglichkeiten zum Schwindeln in der deskriptiven Statistik (Angabe eines Mittelwertes oder „geschönte" Diagram-

me), der stochastischen Modellierung (stochastische Unabhängigkeitsannahmen, reale Zufallszahlengeneratoren) oder der schließenden Statistik (Angabe von Vertrauensintervallen).

Die Antwortzeit als Leistungsmaß ist das entscheidende Kriterium bei der Messung der für einen interaktiven Anwender sichtbaren Kommunikationsleistung, während der Durchsatz hier nicht so entscheidend ist. Dafür aber bei den Datentransportoperationen wie Dateitransfer (engl. File Transfer) oder dem Austausch von elektronischer Post (engl. Electronic Mail), wo wesentlich größere Datenmengen ohne Echtzeitanforderungen versandt werden [157]. Die Verfügbarkeit, die im CIM-Bereich eine wesentliche Rolle spielt [174], wird im folgenden nicht weiter betrachtet.

Leistung bezieht sich auf eine pro Zeiteinheit geleistete Auftragsbearbeitung. Eine Zeiteinheit ist eine wohldefinierte Länge auf der Zeitachse, dem Fortschreiten der Zeit aus der Vergangenheit in die Zukunft. Eine Definition der Begriffe *Ereignis* (engl. Event) und *Zeitdauer* (engl. Duration) findet sich in [118], die abhängig vom Zeitbegriff definiert werden:

- Ereignis — ein Vorkommnis in einem Zeitpunkt, d.h. ein Geschehen in einem Moment auf der Zeitachse, das selbst keine Zeit benötigt;
- Zeitdauer — das Zeitintervall zwischen zwei Ereignissen, d.h. ein Abschnitt auf der Zeitachse.

Zur Bestimmung der Lage eines Ereignisses auf der Zeitachse oder der Länge einer Zeitdauer ist ein Zeitmaß nötig. Dazu existieren nach [199] die *astronomische Zeit*, die zur Bestimmung eines Zeitpunktes definiert ist und auf astronomische Beobachtungen gestützt ist. Die *physikalische Zeit* mit der Zeiteinheit *Sekunde* stützt sich auf die Schwingungsfrequenz von Cäsium, weicht jedoch von der astronomischen Zeit ab. Durch Korrektur wird die physikalische Zeit der astronomischen Zeit angepaßt und führt zur Universal Time Coordinated (UTC). Diese UTC ist die Basis für die Bildung von Zeitzonen. Nachteilig an dieser UTC ist jedoch, daß Zeitsprünge beim Nachstellen bestehen.

Zum Messen der Zeit sind Uhren nötig. Eine physikalische Uhr ist ein Gerät, das, basierend auf periodischen Schwingungen eines Quarzkristalls oder eines Atoms, den Fortgang der Zeit mißt. Die *zeitliche Auflösung* (engl. Granularity) dieser physikalischen Uhr ist die Dauer einer Schwingung, d.h. der Zeitabschnitt zwischen zwei Weiterschaltzeitpunkten. Damit kann die physikalische Uhr im Weiterschaltzeitpunkt nur Schwingungsperioden zählen, weshalb Uhren und Zähler als gleichwertig betrachtet werden. Die *Ganggenauigkeit* (engl. Accuracy) einer Uhr wird durch ihre maximale Abweichung von der astronomischen oder physikalischen Zeit angegeben, wobei zwischen kurzzeitiger und langzeitiger Abweichung (engl. Drift) sowie der Frequenzabweichung unterschieden wird.

Während einer Messung wird jedem Ereignis ein Zeitwert (Uhrenstand) zugeordnet, der im folgenden als *Zeitstempel* bezeichnet wird. Treten zwei oder mehr Ereignisse zwischen zwei Weiterschaltzeitpunkten einer physikalischen Uhr auf, so werden sie als zur gleichen Zeit generiert betrachtet. Alle aufeinanderfolgenden Ereignisse einer Messung bilden eine *Ereignisspur* (engl. Event Trace).

2.1.2 Meßansätze

Bei der Messung einer Hardwareaktion muß eine eindeutige Start- und Endebedingung erkannt werden, damit die Bearbeitungsdauer mit einer Zähleinrichtung (einer Uhr oder einem Zähler mit bekannter Zählfrequenz) erfaßt werden kann. Ein Signal zeigt das Vorliegen dieses Zählwertes an, so daß eine Weiterverarbeitung (Anzeige oder statistische Auswertung) stattfinden kann (Bild 2.1). Durch Akkumulieren im Zähler wird die Summenbearbeitungszeit im Meßintervall bestimmt, falls nicht der Zähler mit dem Beginn eines Auftrags neu initialisiert wird.

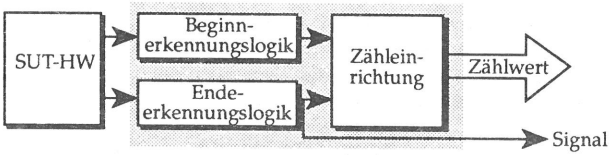


Bild 2.1: Zeitmessung bei einer Hardwareaktion

Die Dauer einer Auftragsbearbeitung durch Programme kann am einfachsten durch Abzählen der notwendigen Befehle und deren jeweilige Ausführungsdauer anhand des erzeugten Programmkodes angegeben werden. [204] beschreibt ein Verfahren, wie durch eine zusätzliche Stufe im Kompilierungsprozeß die notwendige Zyklenzahl für einzelne Blöcke zwischen Verzweigungen automatisch bestimmt und deren Durchlaufhäufigkeit in Tabellen während der Programmlaufzeit gezählt wird (die Auswertung basiert auf dem unmodifizierten Programm, während das instrumentierte Programm bis zum Faktor 2 langsamer wird). Die abgezählte Laufzeit wird aber nur dann korrekt sein, wenn keine Datenzugriffe mit unbestimmter Zugriffszeit enthalten sind. Selbst Programmsegmente mit konstanter Kodelaufzeit können unterschiedliche reale Laufzeiten aufweisen, wenn sie durch andere Kodesequenzen unterbrochen werden. Trotzdem wird die Methode der Kodelängenmessung verwendet, um abgeschätzte Laufzeiteingabewerte für Simulationen und Analysen zu erhalten. Erweitert wurde diese Methode für die programmablaufgesteuerte Simulation (engl. Execution-Driven Simulation), wo die Laufzeiten für einzelne Kodeblöcke als Eingabewerte für Simulationen dienen [194, 195].

Gemessen werden kann die Dauer eines Auftrags auf folgende Arten:

- passives Aufzeichnen aller Systemaktivitäten (engl. Tracing),
- wiederholtes Unterbrechen des Systems und Statusabfrage (engl. Sample-Driven Measurement) oder
- ereignisgesteuertes Melden von Systemaktivitäten (engl. Event-Driven Measurement).

Die erste Methode verlangt, daß alle Punkte, die zur Erkennung von Änderungen der Systemaktivität nötig sind, beobachtet und die dort verfügbaren Informationen gespeichert werden müssen. Hier bieten sich die Schnittstellen eines Objektsystems nach außen an:

- Schnittstelle zum Netz oder
- Schnittstelle zum Anwender.

So definiert die CCITT mit den Empfehlungen X.134 - X.136 [22, 23, 24] Richtlinien zur Messung von paketorientierten Kommunikationssystemen. Die Leistung soll dem zugrundeliegenden Modell nach nur durch Beobachten des Paketverkehrs über sogenannte Referenzereignisse (engl. Primary Layer Reference Events) an Sektionsgrenzen gemessen werden (vergl. Bild 2.2).

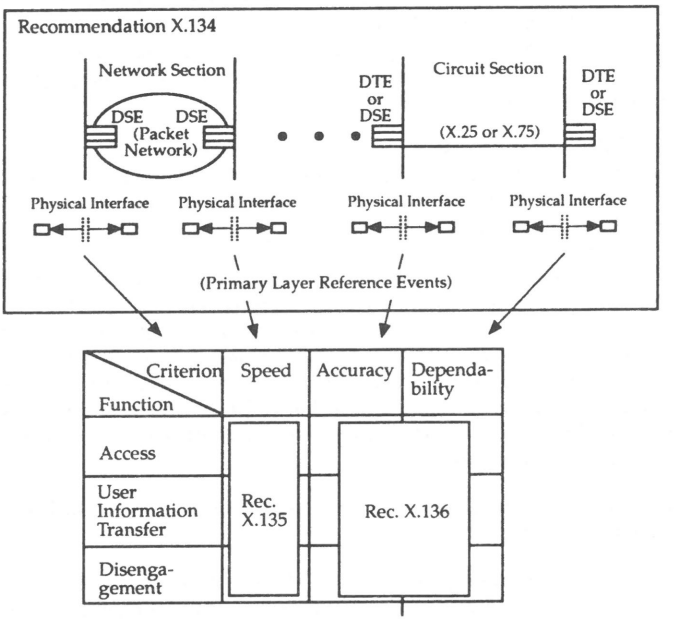


Bild 2.2: Modell zur Leistungsmessung nach CCITT [22, 23, 24, 25]

Derselbe Ansatz wird auch bei der Leistungsmessung im ISDN zugrunde gelegt [20, 21].

Ein weiterer geeigneter, jedoch systeminterner, Beobachtungspunkt ist der Systembus, bestehend aus Datentransferbus, Steuerbus und Adreßbus [183]. Die aufgezeichneten Aktivitäten (insbesondere des Adreßbusses) werden als Eingabe für Simulationswerkzeuge genutzt (engl. Trace-Driven Simulation), um den Einfluß von Caching-Algorithmen [202] oder die Lastverteilung auf Multiprozessorsystemen [76, 113, 115] zu untersuchen. Weiterhin lassen sich die aufgezeichneten Aktivitäten manuell oder programmgesteuert im Anschluß auswerten. Nachteilig an dieser Methode ist, daß meist extrem viel Information erfaßt, gespeichert und danach ausgewertet werden muß. Techniken zur Reduktion der Datenmenge wurden in [202] angegeben. Der Reduzierungsfaktor der Bearbeitungsgeschwindigkeit eines Programmes infolge des Tracing ist in [195] angegeben mit:

- Hardware-Tracing: x 1
- Mikrokodemodifikation: x 10
- Simulation: x 1000

- unterbrechungsgesteuert: x 100
- modifizierte Programme: x 10

Nur beim hardwaremäßigen Erfassen der Information wird das System nicht verändert, d.h. die Informationsdarstellung ist durch die Meßpunkte vorgegeben, was die nachträgliche Interpretation der aufgezeichneten Informationen sehr erschweren kann. Alle anderen Verfahren versuchen, durch spezielle Mikrobefehle oder Programme (durch Unterbrechen nach jedem ausgeführten Befehl (engl. Trap) oder Modifikation des Programms), die zu exportierende Information in kodierter und besser auszuwertender Form zu erzeugen.

Bei der zweiten Methode, der wiederholten Systemzustandsabfrage wie in [36, 134], kann man die Unterbrechungsabstände verändern und damit die zeitliche Auflösung in weiten Grenzen variieren. Die Unterbrechungserzeugung, periodisch oder nicht, kann entweder von außen erfolgen oder über Zeitgeber bzw. real vorhandene Unterbrechungen innerhalb des Systems realisiert sein. Nachteilig ist jedoch der massive Einfluß auf das zu beobachtende Objektsystem, der mit steigender Unterbrechungshäufigkeit zunimmt, andererseits die ungenügende Genauigkeit bei großen Unterbrechungsabständen. Die systemweite Messung eines Systems erfordert bei dieser unterbrechungsgesteuerten Methode, daß alle interessierenden Größen zum Unterbrechungszeitpunkt abgefragt werden, was zu sehr vielen unnötigen Abfragen führen und viel Zeit beanspruchen kann. Damit nun diese Abfragezeit nicht sichtbar wird, wird durch Anhalten der Systemzeit [77] versucht, diese Unterbrechung unsichtbar zu machen. Finden jedoch noch parallele Vorgänge wie Zugriffe auf rotierende Massenspeicher oder Interaktionen mit anderen Systemen am Netz statt, so ist das Systemverhalten i.a. ein gänzlich anderes.

Üblicherweise wird eine der beiden genannten Methoden eingesetzt, jedoch sind auch Kombinationen beider Methoden möglich [1].

Bei der dritten Methode, dem ereignisgesteuerten Melden, stoppt das System nicht zur Bestimmung des Auftragsstatus, sondern liefert nur bei Systemzustandsänderungen ein Ereignissignal. Damit gibt es 3 Sichtweisen, wie Leistung mit dieser Methode gemessen werden kann:

(a) Ereignis identifiziert einen Zustand in einer Folge zur Auftragsbearbeitung;

(b) Ereignisse, die Zustände innerhalb einer Auftragsbearbeitung eingrenzen;

(c) Ereignisse, die Zustände eingrenzen samt Zustandsänderungsbedingung.

Bei der Sichtweise (a) wird die gesamte Auftragsbearbeitung in einzelne Zustände unterteilt [46], die in gewünschter Folge durchlaufen werden.

Gemäß der Methode (b) werden Bearbeitungszustände von Ereignissen eingegrenzt, deren Auftreten als Beginn- bzw. als Endezeitpunkt für die Zeitmessung dienen [197]. Sind alle Aufträge mit Beginn- und Endeereignissen versehen, so können auch Unterbrechungen von Aufträgen durch weitere Aufträge erkannt werden. Problematisch sind hier Unterbrechungen durch asynchrone Ereignisse wie Interrupts oder Prozeßwechsel, die eine Instrumentierung aller Systemroutinen (engl. System Calls) verlangen.

Diese Sichtweise wird mit (c) erweitert, indem zusätzlich zu den einzelnen Zuständen auch die Ursachen für den Zustandswechsel erfaßt werden [135]. Die Zustände werden in einzelne atomare Phasen unterteilt, die, im Gegensatz zu den Zuständen von (a) und (b), nicht unterbrechbar sind.

Dies erlaubt die exakte zeitliche Erfassung von Zustandsdauern basierend auf mehreren Phasen auch in Mehrprogramm-/Mehrbenutzersystemen. Die im Ereigniszeitpunkt miterfaßte Zustandsänderungsbedingung erlaubt im nachhinein das Ablaufgeschehen nachzuvollziehen.

All diesen Sichtweisen ist gemeinsam, daß ein Zustand zeitlich erfaßt werden kann. Bei (a) werden Zustandsdauern gemessen und zur Gesamtbearbeitungszeit zusammengerechnet, während in den beiden anderen Verfahren durch zueinandergehörende Ereignisse (einem eindeutigen Beginn- und einem Endezeitpunkt) die Zustandswechsel markiert werden, wodurch die Bestimmung der Zustandsdauern möglich wird. Mittels geeigneter Meßmittel (Identifizieren der Ereignisse, Berechnen der jeweiligen Zustandsdauern und abschließendes Aufsummieren zur Bearbeitungszeit) können dann die Zeiten zur Auftragsbearbeitung bestimmt werden. Dieser Berechnungsschritt ist bei Sichtweise (c) aufwendiger, da Ereignisse bei Zustandsbeginn, -unterbrechung, -wiedereintritt und -ende erzeugt werden und ein Zustand aus beliebig vielen atomaren Phasen zusammengesetzt sein kann. Ein Problem der Auswertung über Beginn- und Endeereignis besteht darin, daß in der Spur die zusammengehörenden Ereignisse zu finden sind.

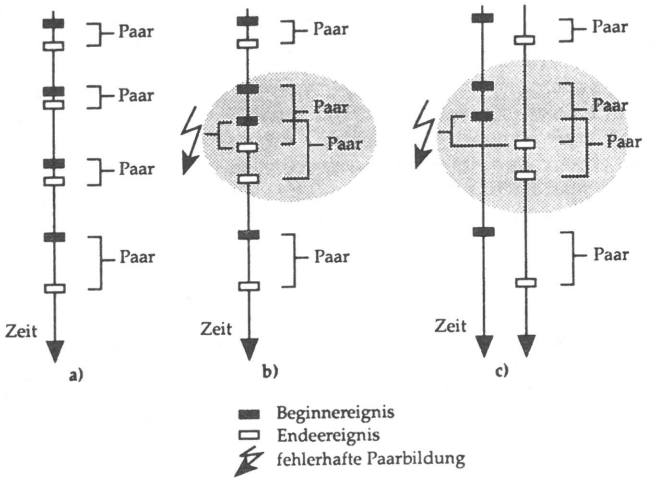


Bild 2.3: Zuordnung von Ereignissen zu Paaren: a) Paare in Reihe, b) Paare verschachtelt, c) zwei Ereignisspuren mit verschachtelten Paaren

Die Vorgabe der Ereignisse zu einem Paar allein reicht oft nicht, so daß weitere Hilfen erforderlich sind. [42] verwendet neben der Ereigniskennung ein Zeitintervall, innerhalb dessen das Endeereignis auf das Beginnereignis folgen muß. Jedes zu früh oder verspätet erkannte Endeereignis wird verworfen. Bei Ereignissen, die nicht immer paarweise auftreten (wie in Bild 2.3a), versagt dieses Verfahren. Diese Situation tritt aber im verteilten Fall bei zwei Ereignisspuren häufig auf, da die beteiligten Objektsysteme meist unterschiedlich schnell sind und nicht synchron arbeiten (Bild 2.3c). Die starre Paarbildung durch Abzählen kann hier helfen, setzt aber voraus, daß nicht innerhalb eines Paares begonnen und alle Ereignisse korrekt erkannt und gespeichert wurden.

Die letzte Form (Paare von Zustandsänderungsereignissen) wird in vielen Leistungsmessungen an der Anwenderschnittstelle verwendet, indem man die Dauer vom Anstoß eines Auftrags bis zu dessen Ende abzählt (vergl. Kap. 2.1.1). Auf diese Weise gewinnt man Leistungswerte allein durch Beobachtung an einem Meßpunkt. Problematisch wird dies in den Fällen, wo neben dem lokalen System weitere Systeme in den Auftrag miteinbezogen sind (vergl. Kap. 3.2.2): in den veröffentlichten Meßergebnissen von [Supplement zu 23, 45, 146, 158] wird die Einwegverzögerung durch einfaches Halbieren der Zeit für Hin- und Rückweg bestimmt, was nicht immer zu gelten braucht. So wird in [53] ein Fall beschrieben, wo das Laufzeitverhältnis 1,5:7 und in [32] das Kodelängenverhältnis 421:448 von Hin- zu Rückweg war.

Außer zur Zustandserkennung und damit zur Zustandsdauermessung kann ein Ereignis auch zur Bestimmung weiterer Leistungsgrößen verwendet werden. Dies erfordert die Extrahierung weitergehender systeminterner Information an dieser Stelle. Mit Hilfe dieser Information, die nur *einem* Erfassungszeitpunkt zugeordnet ist, kann eine weitere Form von Leistungsangabe erzielt werden. Durch Interpretation kann ein Rückschluß auf das systeminterne Verhalten gewonnen werden. Zum einen dokumentiert die erfaßte Information einen momentanen Wert oder Zustand, zum anderen kann durch Verfolgen desselben Meßpunktes die Historie dieses Informationswertes aufgezeichnet werden. So wird aus der momentanen Prozeßkennung die Abfolge der jeweils im Meßpunkt aktiven Prozesse erkennbar. Die aktuelle Länge einer Auftragswarteschlange dient zur Pufferdimensionierung oder kann bei mittelfristiger Verfolgung der Belegungslänge Aussagen über zukünftige Systemengpässe ermöglichen [9, 73]. Die Restzeit eines rückgesetzten Zeitgebers oder die momentane Sendefenstergröße geben zusätzlich Anhaltspunkte für die Systemkonfiguration. Repräsentiert der übernommene Informationswert den aktuellen Zustand eines systeminternen (Aufwärts-)Zählers, so kann durch Differenzbildung zweier Werte in aufeinanderfolgenden Meßpunkten ein abgeleiteter Leistungswert gebildet werden.

2.1.3 Meßpunkte

Leistungsmessungen auf der Basis von Ereignissen, die Zustandswechsel von Systemaktionen anzeigen, werden als ereignisgesteuerte Messungen bezeichnet. Dabei spielen die Meßpunkte, d.h. die Stellen, an denen ein Ereignis signalisiert werden muß, in zweierlei Hinsicht eine entscheidende Rolle: die Plazierung des Meßpunktes selbst und der mit diesem Meßpunkt verbundene Informationsgewinn.

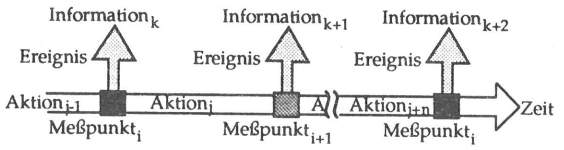


Bild 2.4: Ausgeführte Aktionen, begrenzende Meßpunkte sowie exportierte Informationen

Generell ist es wünschenswert, möglichst viel Information pro Meßpunkt zu erhalten.

Bei allen bisherigen Überlegungen wurde angenommen, daß ein Meßpunkt idealerweise keine Laufzeitbeeinflussung darstellt. Diese Annahme kann beim passiven Beobachten gültig sein, so-

fern die Anschlüsse an das Objektsystem „hochohmig" sind: zusätzliche Abgriffe stellen eine ohmsche und kapazitive Belastung für die Signalleitungen dar. Besonders der Abgriff an Taktleitungen oder hochbelasteten Steuerleitungen ist sehr kritisch und verlangt aufwendige Eingangsschaltungen, die bei schnellen Signalen auf den Leitungswiderstand abgestimmt sein müssen. Da die Erkennung eines Meßpunktes durch Hardwareteile erfolgt, ist auch der zeitliche Erfassungspunkt, abgesehen von minimalen Schwankungen durch Jitter oder thermische Einflüsse, immer derselbe.

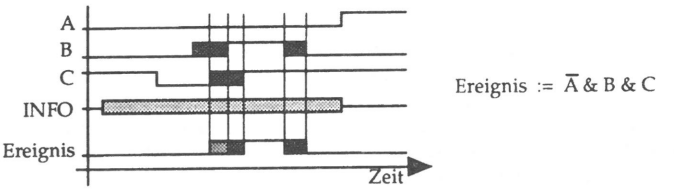


Bild 2.5: Zuordnung von Auftrittszeitpunkt zu Hardware-Meßpunkt

Im einfachsten Fall genügt eine Kombination von Signalzuständen zur Erkennung eines Ereignisses (Bild 2.5). Ist die Kombination von Signalzuständen nicht eindeutig, so sind weitere Zustandsvergleiche zur eindeutigen Identifizierung eines Meßpunktes nötig. Es entsteht eine Sequenz von Signalzuständen, die in Folge einen Meßpunkt bilden und mittels eines Hardware-Sequenzers erkennbar sind [7]. Wird die vorgegebene Folge nicht eingehalten, so setzt man den Sequenzer zurück und beginnt erneut mit der Erkennung des ersten Signalzustandes. Die zeitlichen Abstände zwischen einzelnen Signalzuständen müssen dabei nicht äquidistant sein. Als Auftrittszeitpunkt wählt man in einer Sequenz den letzten Signalzustand.

Begrenzen die Ereignisse eine in Hardware ablaufende Aktion, so verwendet man an diesem Meßpunkt üblicherweise immer dieselbe Informationskodierung mit festen Feldern aber unterschiedlichen Informationsinhalten.

Durch Softwareaktionen erzeugte Meßpunkte benötigen Befehle zur Informationsbereitstellung und spezielle Befehle zur Ereignisgenerierung. Die dafür erforderliche Zeit, die abhängig von der Informationsmenge, der Informationskodierung und der verwendeten Form der Ereignisgenerierung ist, beträgt ein Vielfaches einer Befehlsbearbeitungszeit. Üblicherweise wird das Ereignis erst

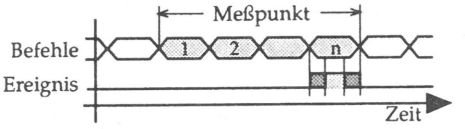


Bild 2.6: Zuordnung von Auftrittszeitpunkt zu Software-Meßpunkt

am Ende der Meßpunktbefehlssequenz, d.h. nach Zusammenfügen der Meßpunktinformation, signalisiert. Ein Software-Meßpunkt wird letztendlich durch Hardwaremittel erkannt und ist damit ein Software-initiierter Hardware-Meßpunkt. Der Zeitpunkt der Ereignisgenerierung wird demzufolge auch durch die befehlsinterne Bearbeitungsweise (Befehl *n* in Bild 2.6) beeinflußt.

Wie beim Hardware-Meßpunkt kann eine Sequenz von Zuständen einen Meßpunkt bilden, d.h. daß viele Befehle zur Informationsbereitstellung und zur Speicherung erforderlich sind. Der Auftrittszeitpunkt dieses komplexen Meßpunktes kann unterschiedlich zugeordnet werden:

- die Sequenz selbst liefert den Zeitstempel (z.B. durch Auslesen einer internen Uhr),
- nur der letzte Abspeicherungsbefehl wird markiert oder
- jeder einzelne Abspeicherungsbefehl wird zeitlich markiert.

Im ersten Fall ermittelt man die Auftrittszeit innerhalb der Sequenz im Objektsystem, was zu weiteren Systembeeinträchtigungen und einer geringen zeitlichen Genauigkeit führt. Entsprechendes gilt bei Markierung des letzten Befehls. Hier sollte die Laufzeit der Programmsequenz am Meßpunkt kurz sein, damit der Zeitstempel auch für die ersten gespeicherten Informationen gültig ist. Speichert man zu jedem Informationswert den Zeitstempel mit ab, so kann sogar die Laufzeit zwischen den einzelnen Speichereinträgen bewertet werden.

Ein Einfluß auf das Objektsystem ist bei allen Software-Meßpunkten meßbar. Selbst wenige Programmschritte stellen eine Veränderung des Systems dar, da auf unterschiedliche Objektsystemvariablen zugegriffen werden muß:

- Register sind in Ein-/Ausgabegeräten oft nur beschreibbar, weshalb eine lokale Kopie dieser Registerinhalte nötig ist,
- Register, die lesbar sind, werden meist nach einem Lesezugriff verändert, so daß auch hier Kopien für spätere Zugriffe gehalten werden müssen,
- der Zugriff auf Speicherzellen zur Informationsbeschaffung ist nicht unkritisch, da dies in Systemen mit Pufferspeichern (engl. Cache Memory) oder Kachelspeichern (engl. Paged Memory) zu Fehlzugriffen in einem der Speicher und damit zum Verändern des Systemzustandes führen kann.

Diese Systembeeinflussung kann durch eine sorgfältige Auswahl der relevanten Information und eine Kodierung deutlich reduziert werden. Weiterhin ist eine dynamische Anpassung der Informationsmenge möglich, führt jedoch zu unterschiedlichen Laufzeitbeeinflussungen. Als noch tolerierbare Obergrenze wird in [135] und [176] ein Laufzeitfehler von 5% angegeben, bei Echtzeitsystemen (engl. Real-Time Systems) wird selbst dieser kleine Fehler nicht mehr toleriert [155, 200]. Diese 5%-Fehlerschranke gilt nur für Einprozessorsysteme, in Mehr- oder Multiprozessorsystemen kann dies aufgrund der Synchronisationsmechanismen zu gänzlich anderem Systemverhalten führen. [195] gibt an, daß Systeme mit gemeinsamem Speicher (engl. Shared Memory) empfindlicher auf moderate Veränderungen reagieren als Systeme ohne Speicherkopplung.

Müssen trotzdem viele Informationen an den Meßpunkten extrahiert werden, so ist es günstiger, mehrere Messungen mit jeweils unterschiedlichen Meßpunkten zu machen, um den Laufzeitfehler so klein wie möglich zu halten. Die Form der extrahierten Information hängt von der Informationsmenge ab und kann entweder mit einem Ereigniswort oder, bei größerer Informationsmenge, als *Ereignisstruktur* (engl. Event Record) dargestellt werden.

Wird nur der Meßpunkt selbst erkannt, so ist wenig Einsicht in die systeminternen Entscheidungsvorgänge möglich. Durch die Menge der an diesem Meßpunkt gewünschten Information als auch durch die dabei tolerierbaren Systembeeinflussungen wird das einzusetzende Meßverfahren festge-

legt. Hierbei gibt es verschiedene Stufen, die die Sichtweite der erreichbaren Informationen definieren und auf der Ebene von

- Mikroprogramm-Befehlen,
- Maschinenprogramm-Befehlen,
- Hochsprachen-Zuweisungen,
- Blockstrukturen,
- Programmen,
- Prozessen oder
- Systemen

liegen.

Leistungsmessungen auf der Mikroprogramm- und Maschinenprogramm-Ebene liefern Informationen mit bester zeitlicher Auflösung. Beide werden vorwiegend zur Laufzeitmessung der einzelnen Befehle [155] und der Optimierung von Befehlssequenzen [167] eingesetzt. Messungen an Hochsprachen-Zuweisungen sind interessant bei der Compilerentwicklung, aber auch wenn es um den Laufzeitfehler durch zusätzlich eingefügte Zuweisungen [142] geht. Bereits auf dieser Ebene kann eine Optimierung von kritischen Pfaden (z.B. durch geänderte Kodierung) erfolgen. Blockstrukturen lassen sich bei Hochsprachen sehr einfach mit den Strukturierungselementen Funktion und Prozedur bilden. Damit können Meßpunkte identisch sein mit den Aufruf- und Rückkehrpunkten [181]. Aber auch Teile von Funktionen oder Prozeduren stellen Blöcke dar, deren Blockgrenzen durch spezielle Meßpunkte gebildet werden [211]. Die nächsthöhere Ebene erlaubt Messungen auf Prozeß-Ebene [200]. Damit läßt sich auch innerhalb von Betriebssystemen das Zusammenspiel von einzelnen Systemprogrammen mit Anwenderprozessen beobachten und messen [64, 137]. Die Programm-Ebene entspricht der klassischen Leistungsmessung mittels unterschiedlichen Bewertungsprogrammen, bei denen entweder die Programmbearbeitungsdauer relativ zu einer Referenzmaschine oder die in vorgegebener Zeit ausgeführten Aufträge als Leistungsmaß angegeben werden [203]. Messungen auf System-Ebene beziehen die Applikationen und das Benutzerverhalten in die Messung mit ein. Als Meßpunkt bietet sich hier besonders die Schnittstelle zu Systemroutinen an [154].

Je nach Ebene, auf der Messungen durchgeführt werden sollen, muß die Erkennung eines Meßpunktes und die anschließende Erfassung von identifizierender Information mit entsprechender Zeitauflösung erfolgen (Tabelle 2), die deutlich über der minimalen Dauer einer Aktion dieser Ebene liegen muß .

Daneben ist es von essentieller Bedeutung, signifikante Stellen bei der Auftragsbearbeitung zu identifizieren, um daraus ein Modell zu entwickeln. Orientiert man sich dabei an der Implementierung des Problems, z.B. am Blockdiagramm eines Softwareprogramms, so erhält man ein Strukturmodell. Hier lassen sich sehr einfach durch Analyse der Bearbeitungsschritte wesentliche Meß-

Ebene	Zeitauflösung	typische Anwendung
Mikroprogramm	1 ns	Mikrokode-Optimierung
Maschinenprogramm	10 ns	Maschinenkode-Optimierung
Hochsprache	100 ns	Optimierung eines Kodegenerators
Block	100ns...1ms	Laufzeitmessung
Programm	1ms...1s	Laufzeitmessung
Prozeß	1ms...1s	Prozeßverwaltung im Betriebssystem
System	1ms...1s	interaktive Anwendungen

Tabelle 2: Meßebenen und deren zeitliche Auflösung

punkte identifizieren. Geht man jedoch von den Zuständen aus, die zur Abarbeitung eines Auftrags nötig sind, so gelangt man zu einem Funktionsmodell. Dieses Modell kann auch aus der Kenntnis des Bearbeitungsablaufs heraus in Form eines Flußmodells (engl. Flow Chart Model) entwickelt werden. Eine weitere Modellform stellen die Leistungsmodelle dar, die zur analytischen und simulativen Leistungsbewertung verwendet werden. Bei diesen Modellen werden, ausgehend vom Systemverständnis, Annahmen für die einzelnen Modellkomponenten gemacht. Eine Variante stellt das empirische Modell dar, bei dem durch Skalierung das Modellverhalten dem realer Systeme angepaßt wird.

Die Stelle, an der Meßpunkte plaziert werden können, hängt stark von den verwendeten Meßverfahren ab. Trotzdem bieten sich Punkte an, an denen die Systemleistung meßbar ist:

- Schnittstelle zwischen Objektsystem und Netz,
- Schnittstelle zwischen Anwendung und Objektsystem und
- einem beliebigen Punkt im System (sogar im Betriebssystem selbst).

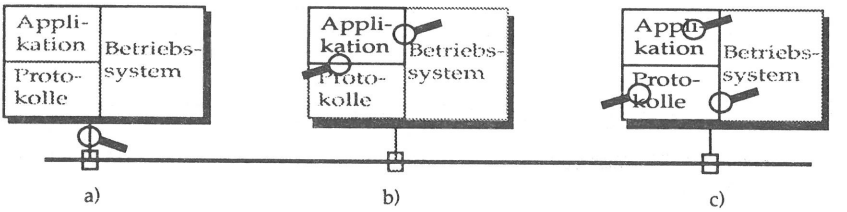


Bild 2.7: Meßpunktplazierung: a.) Netzwerkschnittstelle, b.) Schnittstelle zur Anwendung, c.) im gesamten System

Tabelle 1 ordnet Publikationen, die über Messungen an Kommunikationssystemen berichten, entsprechend diesen drei Stellen zur Meßpunktplazierung ein. Dabei fällt auf, daß die meisten Messungen an der Schnittstelle zum Anwender leere Netze und Systeme sowie synthetische Applikationen (meist fiktiver Art) fordern, um nicht quantifizierbare Einflüsse aufgrund von externem Netzverkehr oder stationsinterne Interferenzen mit weiteren Kommunikationsdiensten zu vermeiden.

Je nach gewähltem Punkt stellt sich der Rest des Systems als undurchsichtige Einheit (engl. Black Box) dar. Im Falle der Schnittstelle zwischen Anwender und System bleibt das gesamte Verhalten innerhalb des Systems, d.h. Betriebssystem, weiteren Anwendungen, Protokollbearbeitung und Nachrichtenverkehr mit dem Netz, unsichtbar. Wird der Netzzugangspunkt gewählt, so überlagern sich alle Anwendungen zu einem Strom von Nachrichten, die über das Netz auszutauschen sind, während alle system- und protokollinternen Abläufe unsichtbar bleiben. Einzig eine Plazierung innerhalb des Systems und des Protokollstacks an den signifikanten Stellen erlaubt die Erkennung der Einflüsse einzelner Anwendungen und die Zuordnung dieser zum überlagerten Netzverkehr.

2.1.4 Meßverfahren

Ist der Meßpunkt bekannt, so bleibt noch offen, wie die interessierende Information sowie die Zeitinformation, die diesem Auftreten des Meßpunktes zugeordnet werden muß, erfaßt, gespeichert und weiterbearbeitet werden kann.

Die bekannten Verfahren lassen sich einteilen in:

- Accounting,
- Hardware-Monitoring,
- Software-Monitoring und
- Hybrid-Monitoring.

Accounting ist ein nahezu in jedem System vorhandenes Mittel zur Leistungsmessung. Dabei wird auf die im System intern erstellten Abrechnungsinformationen zurückgegriffen. Üblicherweise werden Systemaktivitäten bei deren Ausführung, d.h. Aufruf und Terminierung, erkannt und in systeminternen Tabellen gezählt. Ein Einfluß auf die erzeugbare Information besteht meist nicht. Als Zeitgeber dient entweder ein Ticker, der mit jeder periodischen Unterbrechung inkrementiert wird, oder eine rechnerinterne Uhr. Die damit erzielbare zeitliche Auflösung ist systemabhängig und niedrig (im Bereich weniger Millisekunden), wodurch nur Bewertungen von der Block-Ebene aufwärts möglich sind.

Monitore sind viel mächtigere Werkzeuge. Die Aufgabe eines Monitors liegt in der Erfassung von Information in Meßpunkten mit möglichst minimaler Beeinflussung des Objektsystems. Dabei reicht die Bandbreite vom Hardware-Monitor, bei dem durch hochohmig angeschlossene Meßfühler die Meßpunkte durch dedizierte Schaltungen oder Meßgeräte wie Logikanalysatoren erkannt werden, bis hin zu den Software-Monitoren, die beim Durchlaufen eines Meßpunktes die gewünschte Information durch eine Monitorprozedur aus dem System extrahieren und mit dem Auftrittszeitpunkt abspeichern. Beide Monitorvarianten haben Nachteile, die man mit einer Kombination beider Varianten, dem Hybrid-Monitor, zu verhindern sucht.

Wesentlich für die Art der möglichen Auswertungen ist die Frage der Meßinformationsspeicherung. Werden während der Meßdauer alle erfaßten Informationen erst in einem Pufferspeicher abgelegt, so kann die Auswertung der Information nur nach der Messung (engl. Off-Line) erfolgen. Durch Einsatz eines Zweitorspeichers (engl. Dual Ported Memory), bei dem über ein Tor die Informationen eingeschrieben und über das zweite Tor parallel oder quasi-parallel ausgelesen werden können, kann eine schritthaltende Auswertung (engl. On-Line) realisiert werden. Nachteilig ist jedoch die im Speicher selbst notwendige doppelte Bandbreite, da zwei Zugriffe (Schreiben und

Lesen), während der Meßdauer möglich sein müssen. Der Zweitorspeicher kann dabei in zwei Modi betrieben werden:

- Einschreiben bis Speicherende erreicht ist oder
- Anlegen eines logischen Ringpuffers.

Beim Einschreiben bis Speicherende ist die Aufzeichnungslänge bekannt und endlich. In Abhängigkeit der Einschreibfrequenz ergibt sich die maximale Meßdauer, die nur verkürzt werden kann.

Wird jedoch der Zweitorspeicher als Ringpuffer organisiert, so hängt die virtuelle Länge L_{virt} nur vom Verhältnis der Einschreibfrequenz F_{Ein} zur Auslesefrequenz F_{Aus} und der realen Länge des Speichers L_{real} ab:

$$L_{virt} = \left\lfloor \frac{F_{Ein}}{(F_{Ein} - F_{Aus})} * L_{real} \right\rfloor \qquad \text{(Gl. 1)}$$

Sind Einschreib- und Auslesefrequenz gleich, so erscheint der Speicher unendlich groß. Bei dieser Speicherorganisation kann die Auswertung beginnen, während die Messung noch läuft [201].

Ist eine Zwischenspeicherung nicht nötig oder erwünscht, so muß die Auswertung der Information zum Auftrittszeitpunkt garantiert sein, d.h. in der Zeit zwischen zwei eintreffenden Informationswerten muß die Erkennung, Dekodierung, Auswertung und möglicherweise eine Anzeige erfolgen. Dabei kann eine Geschwindigkeitsanpassung zwischen Objektsystem und Auswertungseinheit nötig sein. Zu diesem Zweck werden FIFO-Bauelemente (engl. First-In First-Out) zur Entkopplung eingefügt. Kommerzielle FIFO-Bauelemente haben deutlich mehr als einen Speicherplatz, so daß obige Gleichung (Gl.1) auch hier für die virtuelle FIFO-Länge gilt.

Verwendet man FIFO-Speicher zur Geschwindigkeitsanpassung, dann kann die Zuordnung eines Zeitstempels zu dem Informationswert vor Einschreiben in oder nach Auslesen aus dem FIFO-Element erfolgen. Zuordnen beim Einschreiben hat den Vorteil, daß der Auftrittszeitpunkt genau erfaßt wird und damit unabhängig von der momentanen FIFO-Belegung ist, jedoch der Zeitstempel im FIFO-Element mitgeführt werden muß. Die zeitliche Zuordnung beim Auslesen aus dem FIFO-Element reduziert dagegen die erforderliche Breite des FIFO-Elementes auf Kosten der Zeitgenauigkeit infolge unbekannter Belegungslänge im FIFO-Element.

Eine Forderung bei der Speicherfrage ist, ob Online- oder Offline-Auswertungen machbar sein sollen. Für Auswertungsverfahren ist es im allgemeinen einfacher, wenn alle Informationen gespeichert vorliegen. Dann kann auf die Informationen im Speicher mehrmals zugegriffen werden, wie es bei der Quantil-Berechnung oder beim adaptiven Aufbau einer Klassenverteilung nötig ist: im ersten Durchlauf werden die minimalen und maximalen Werte und die daraus resultierenden Klassenbreiten ermittelt, bevor im zweiten Durchlauf die Informationen aus dem Speicher in die entsprechenden Klassen einsortiert werden. Beim Online-Auswerten sind die Informationen aber nur einmal verfügbar.

2.2 Meßkonzepte

Obwohl das Ziel einer Leistungsmessung vorgegeben ist, gibt es noch keine allgemeine wissenschaftliche Methode, wie Leistungsmessung durchzuführen ist. Die grundlegende Frage, was gemessen werden soll und wozu die Resultate dienen sollen [152], wird beantwortet durch die Aussage:

„The key to performance evaluation as well as to system design is to understand what systems are and how they work" [14]

und das Zitat von R.W.Hamming:

„The purpose of measurement is insight not numbers" [in 15].

In [6] wird eine Methode zur Leistungsverbesserung vorgeschlagen, deren erster Schritt das Systemverständnis, basierend auf Leistungsmessungen, voraussetzt.

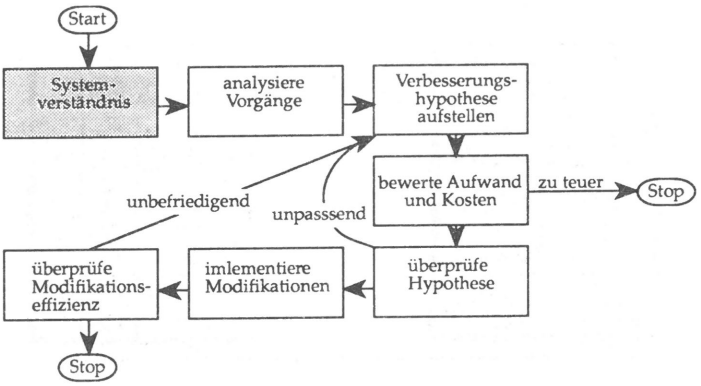


Bild 2.8: Vorschlag einer Leistungsverbesserungsmethode nach [6]

Als Mittel zum Erwerb des Systemverständnisses wurden in [46, 197, 135] unterschiedliche praktisch einsetzbare Meßansätze beschrieben, die alle einen Monitor benötigen. Die Interpretation der gemessenen Daten bleibt dem Benutzer überlassen. Andererseits gibt es standardisierte Ansätze zur Systemverwaltung (engl. System Management) von der ISO [96, 102], wo die zur Leistungsmessung [106] benötigten Informationen [107] sowie die zugehörigen Zugriffsdienste (Common Management Information Services, CMIS) [100] und Protokolle (Common Management Information Protocol, CMIP) [101] definiert sind. Auf dieses Managementmodell bezieht sich auch MAP/-TOP [130]. Darin sind jedoch keine Richtlinien zur Meßwerterfassung angegeben, vielmehr wird die Art, wie die interessierenden Informationen (bei der ISO als Objekte bezeichnet) zu ermitteln sind, dem Implementierer überlassen. Weitere Ansätze zur Systemverwaltung [18] basieren auf ähnlichen Architekturen, d.h. einer Aufspaltung in Datenbasis mit Verwaltungsdaten (engl. Management Information Base, MIB) und Zugriffsfunktionen, etwa im Internet mit seinem Modell einer einfachen MIB [133] und dem Protokoll SNMP (Simple Network Management Protocol) [17]. In

dieser Datenbasis sind für jede Protokollschicht Objekte definiert, es können aber eigene Objekte hinzugefügt werden. Wie die Informationswerte gewonnen werden, ist auch hier nicht festgelegt.

Im folgenden werden die Ansätze der offenen Systeme zur Leistungsmessung untersucht und auf ihre Eignung im Hinblick auf das vorliegende Meßproblem bewertet.

2.2.1 ISO OSI-Leistungsmessung

Als wesentliche Komponenten im ISO/OSI-Managementmodell gibt es Managementprozesse, die entsprechend ihrer Rolle als steuernder (Manager) oder gesteuerter Partner (Agent) die Bearbeitung der Objekte zulassen. Objekte bezeichnen bei ISO die Managementsicht auf Netzressourcen. Die eigentliche Information ist in den Attributen der Objekte enthalten. Attribute speichern Zustände, Leistungsgrößen und Konfigurationsparameter. Neben den Attributen sind für ein Objekt Ereignismeldungen und Zugriffsmöglichkeiten definierbar. Der Manager kann über den Agent mit Hilfe des Protokolls CMIP, das den 7-schichtigen Kommunikationsstack benutzt, auf die Objekte zugreifen.

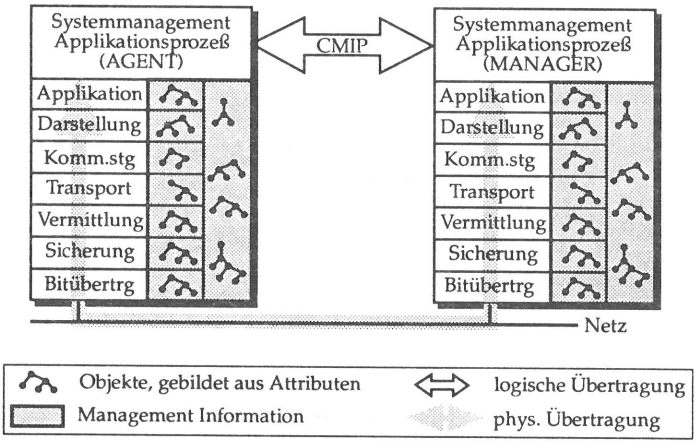


Bild 2.9: ISO/OSI-Managementmodell [102]

ISO 7498 [96] teilt den Management-Aufgabenbereich in 5 Unterbereiche auf:

- Fault Management,
- Accounting Management,
- Configuration and Name Management,
- Performance Management und
- Security Management.

Die einzelnen Aufgaben des *Performance Management* werden unterstützt durch System-Management-Funktionen, die in den Standards ISO10164-1 bis ISO10164-13 näher beschrieben sind. Daneben wird dieser Managementbereich entsprechend den Zielen strukturiert:

- Extrahierung von Information (engl. Monitoring),
- Meldungserstellung von Komponenten (engl. Reporting) und
- Steuerung von Komponenten (engl. Controlling).

So sollen gemäß der *Alarm Reporting Function* [103] Alarme ausgelöst werden, wenn fehlerandeutende Situationen in einem zu kontrollierenden System auftreten. Darunter versteht man Ausfälle oder Fehlverhalten von Komponenten, Fehler bei der Bearbeitung der Kommunikationsaufgaben, Verstöße gegen die Dienstgüte oder äußere Einflüsse, die ihrer Schwere nach klassifiziert sind.

Durch die *Event Reporting Function* [104] und *Log Control Function* [105] werden die von Objekten erzeugten Ereignismeldungen (engl. Notifications) entweder in Form von Event Reports weitergeleitet oder abgespeichert.

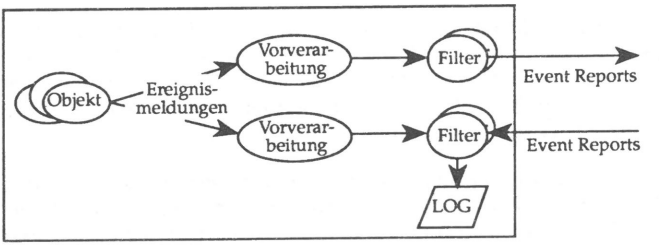


Bild 2.10: Ereignisbearbeitung nach ISO10164 [104, 105]

Die *Workload Monitoring Function* [106] formuliert als Ziel, die Güte der Kommunikationsabwicklung zu überwachen. Drei Modelle helfen dabei, die Belastung der Kommunikationssysteme zu bewerten:

- Resource Utilization Model:	gibt Aufschluß über die Belastung der OSI Betriebsmittel;
- Rejection Rate Model:	dient der Betrachtung der Rate von zurückgewiesenen Diensteanforderungen;
- Resource Request Rate Model:	modelliert die Rate der Anforderung von OSI Betriebsmitteln.

Metric Objects (abgeleitet aus Objekten) eines lokalen oder entfernten Systems stellen die Basis dieser Modelle dar. Durch periodisches Lesen von Bereichsanzeigen oder Zählern wird die Einhaltung eines erlaubten Bereichs überprüft, ansonsten erfolgt die Benachrichtigung des Managerprozesses. Beim sogenannten *Mean Monitor* genügt eine Abweichung vom berechneten Mittelwert,

um eine Ereignismeldung in Form eines QOS-Alarms (engl. Quality Of Service, QOS) zu veranlassen.

In [106] sind für die Leistungsmessung die grundlegenden Typen der Attribute angegeben:

- Counter: einfaches Zählerelement,
- Gauge: Bereichsanzeige,
- Threshold: Schwellwert, (einem Zähler oder einer Bereichsanzeige zugeordnet) und
- Tide Mark: Speicheranzeige der Extremwerte einer Bereichsanzeige.

Im Falle der *Counter* gibt es zwei Realisierungsmöglichkeiten. Die erste Variante ist nicht initialisierbar, sondern wird zu Beginn auf Null gesetzt und zählt bei jeder zugeordneten Aktion um einen Zählerstand weiter. Die alternative Variante kann jederzeit gesetzt oder gelöscht werden. Beide Zählerformen können beim Zählerüberlauf ein Ereignis auslösen. Ein *Gauge*-Wert pendelt zwischen dem minimalen und maximalen Bereichswert. *Threshold counter* sind spezielle Zähler, die beim Erreichen des Schwellwertes oder nach einer gewissen Anzahl von Zählerinkrementierungen ein Ereignis auslösen. Dabei lassen sich einer Aktion auch mehrere Schwellwerte zuordnen. *Tide Marks* speichern die jeweiligen Grenzwerte einer Bereichsanzeige. Zum Zugriff auf Attribute gibt es die zwei erlaubten Operationen Lesen (GET) und Schreiben (SET).

Einzelne Attribute bilden so die Information für ein schichtenspezifisches oder schichtenübergreifendes Objekt (vergl. Bild 2.9). Alle Objekte zusammen stellen eine Datenbasis für Verwaltungszwecke dar. Der Zugriff auf die MIB-Objekte wird über das CMIP abgewickelt, bei dem ein Partner den steuernden Part übernimmt. Zusätzlich sind Managementprotokolle zwischen äquivalenten Schichten standardisiert, die jedoch unabhängig von der MIB sind.

Bisher waren Inhalt der Managementstandards bei ISO nur Objekte, die auf Attributen innerhalb der Kommunikationsschichten aufbauen, d.h. Attribute zur Messung von CPU-Auslastung oder Speicherbelegung fehlen. Und da die bereits standardisierten Attribute auf Zählern basieren, kann die Zeitmessung nur durch Mittelwertbildung innerhalb eines Abfrageintervalls gebildet werden. Es ist möglich, weitere Objekte zu erzeugen, die für die weitergehende Messung einer Rechenanlage nötig erscheinen.

Der Aufwand, der allein zum Performance Management aus ISO/OSI-Sicht nötig wird, ist beträchtlich: die über die einzelnen Attribute meßbaren Daten sind entweder mit Hilfe des Kommunikationsprotokolls CMIP von einem auf dieser Anlage installierten Managementprozeß zu einer zentralen abfragenden Rechenanlage zu übertragen oder in Form von Alarms oder Reports selbständig vom Agenten an den Manager zu verschicken. Dynamische Erzeugung und Zerstörung von Objekten sowie die Objektorientiertheit und die Forderung eines 7-schichtigen Protokollstacks als Grundlage des Meßdatenaustausches erschweren den Einsatz in kleinen Systemen.

2.2.2 Leistungsmessung im Internet

Ganz ähnlich wie beim ISO/OSI-Managementmodell wird zur Systemverwaltung im Internet eine Datenbasis [133] zugrundegelegt. Im Gegensatz zu OSI gibt es keine objektorientierte Sicht. Für jedes Protokoll ist eine Gruppe von Variablen definiert, die in allen Realisierungen implementiert

sein muß. Darüber hinaus gibt es einen Zweig in der Datenbasis (Private Subtree), in den die privaten Organisationen ihre speziellen Objekte legen können.

Über das Kommunikationsprotokoll SNMP [17] muß jedes Objekt separat abgefragt werden. Meldungen (Traps) bleiben auf wenige fatale Fehler begrenzt. Insgesamt ist das verwendete Protokoll SNMP im Vergleich zu CMIP deutlich einfacher. So sind die in einem SNMP-Datagramm transferierbaren Informationsmengen stark begrenzt, weshalb mit dem SNMP-Protokoll größere Objekte sequentiell über die GET-NEXT-Anforderung gelesen werden müssen. Dazu wird auf die Protokolle Internet Protocol (IP) und Transmission Control Protocol (TCP) bzw. User Datagram Protocol (UDP) entsprechend Bild 2.11 aufgebaut.

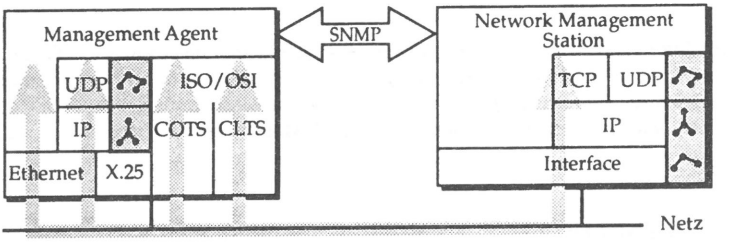


Bild 2.11: Internet Managementmodell

Wesentliche bisherige Einschränkung bei der Datenbasis ist, daß nur für wenige Schichten wie Interface, IP, TCP oder UDP Objekte definiert wurden. Als Vorteil wirkt sich aus, daß nur minimale Anforderungen durch den Agenten und das verbindungslose Protokoll SNMP entstehen: da SNMP direkt auf der Schicht 2 aufsetzen kann, ist es in jedem System einsetzbar und zusätzlich wird, aufgrund der asynchronen Abfrage/Anwort-Betriebsart durch SNMP, der Implementierungsaufwand für einen Managementagenten minimal. Der Arbeitsaufwand bei der Auswertung der Objektwerte muß hier von einer zentralen Managerstation erbracht werden, eine Vorverarbeitung und Filterung findet im Agenten nicht statt.

Objekte speziell zur Leistungsmessung sind noch nicht in die MIB aufgenommen. Dafür sind von verschiedenen Herstellern ergänzend im *Private Subtree* Objekte zur geräteabhängigen Leistungsmessung innerhalb des Herstellerzweiges eingeführt worden. Damit können Objekte für beliebige Meßaufgaben im System installiert werden.

2.2.3 Messung mit einem Monitor

Beide Modelle zur Systemverwaltung (ISO/OSI- und Internet Managementmodell) stellen zwar eine Objektdatenbasis und Kommunikationsprotokolle zum Transfer dieser Objektwerte bereit, wie diese in den realen Systemen extrahierbar sind, bleibt undefiniert. Eine alternative Möglichkeit besteht durch den Einsatz eines Monitors:

> Monitoring is the extraction, processing, collection and presentation in a useful and intelligible format of „dynamic" information regarding the operation of a system [190].

Damit eignet sich die Verbindung eines Monitors mit dem Managementmodell von OSI zur Leistungsmessung von vernetzten Systemen [132].

Ein System mit Netzzugang basiert heute üblicherweise auf dedizierten Kommunikationsbausteinen [90], die den physikalischen Anschluß ans Netz unterstützen. Die Zustandsautomaten der darauf aufbauenden Protokollschichten werden dann durch Software-Programme nachgebildet. Abhängig von den technischen Möglichkeiten liegt die Grenze zwischen Hard- und Software an unterschiedlichen Stellen: liegt diese Grenze oberhalb der ISO Schicht 2a im einen System [88, 89], so reicht in einem anderen System [163, 30, 31] die Hardware bereits bis zur Schicht 4. Der Monitor muß Informationen sowohl in den mit Hardwarebauteilen gebildeten Schichten als auch in Programmen extrahieren, wenn eine gesamtheitliche Betrachtung eines vernetzten Systems erfolgen soll.

Ist nur die Informationserfassung in den untersten, in Hardware implementierten, Schichten gefordert, so bietet sich der Hardware-Monitor an. Sind ausschließlich Informationen in den oberen Schichten, den Anwendungen oder dem Betriebssystem erwünscht, so genügt der Software-Monitor. Die Kombination beider Varianten im Hybrid-Monitor erlaubt Messungen im gesamten System.

2.3 Monitore

Die wesentlichen Schritte beim Monitoring sind:

- Meßpunktfestlegung,
- Einbau der Meßpunkte,
- Aktivierung der Meßpunkte,
- Meßphase,
- Festlegung der Auswertung und Anzeige sowie
- Informationsauswertung und Anzeige.

Die ersten beiden Schritte verlangen eine Kenntnis oder ein Modell des zu messenden Systems. Sind die Meßpunkte eingebaut, dann sammelt der Monitor während einer Meßphase die Informationen über aktivierte Meßpunkte. Bei der Offline-Auswertung finden die Schritte Festlegen und Auswerten nach der Messung statt, während beim Online-messen die Festlegung von Auswertung und Anzeige im voraus gemacht werden muß. Die Ablaufreihenfolge dieser sechs Schritte hängt auch von der Unterstützung durch automatisierbare Konzepte ab, etwa wenn durch Meßaufträge (Festlegung der Auswertung) die einzubauenden Meßpunkte und die durchzuführende Meßphase bestimmt werden [190].

Üblicherweise werden die Meßpunkte vom Monitoranwender definiert. Durch Analyse des Objektsystems legt er die Stellen fest, an denen später gemessen werden soll. Automatische Werkzeuge zur Erzeugung von Meßpunkten an Prozeduraufruf- und Rücksprungstellen während des Übersetzungslaufes [181] oder an Verzweigungsstellen eines Programmes [204] nehmen dem Benutzer die Aufgabe des fehleranfälligen Einbaus von Monitorprozeduren ab und generieren gleichzeitig Strukturen, die eine anschließende Auswertung der aufgezeichneten Meßwerte unterstützen. Diese Werkzeuge erzeugen immer identische Modifikationen, die kaum auf das aktuelle Meßproblem an-

gepaßt sind. Will der Benutzer die Stellen, an denen Meßwerte zu erzeugen sind, genau vorgeben, so sind diese Mittel nicht flexibel genug, da immer alle potentiellen Meßpunkte eingebaut werden.

Die Sorgfalt beim Festlegen des Abgriffsortes hat entscheidenden Einfluß auf die Güte der Messung. Sämtliche Daten, die Identifikation des Meßpunktes und die dort eventuell zu extrahierenden Informationen müssen hier erreichbar sein. Eine erste Schwierigkeit entsteht dadurch, daß die optimale Stelle aufgrund mechanischer Gründe (innerhalb von integrierten Schaltungen), physikalischer Begrenzungen (Leitungsbelastung) oder Beschränkungen des Software-Systems (Speicherschutz) keinen Zugriff erlaubt. Abhilfe kann ein spezieller Meßanschluß [197] bringen, der oftmals zu Diagnosezwecken bereits existiert, nun aber als örtlich konzentrierter Abgreifpunkt für Meßdaten verwendet wird [155]. Damit ist eine Änderung des Monitoranschlusses an das Objektsystem im Betrieb realisierbar und der Eingriff in dieses System kontrollierbar.

Weiter kann während des zeitlichen Auftretens des charakteristischen Meßpunktes die ebenfalls in diesem Zeitpunkt zu extrahierende Information nicht unmittelbar verfügbar sein, weshalb entweder passiv darauf gewartet oder diese aktiv aus dem System entnommen werden. In diesen beiden Fällen ist der zugeordnete Zeitpunkt willkürlich festzulegen:

- auf den Zeitpunkt, an dem ein Meßpunkt erkannt wird,
- auf den Zeitpunkt, an dem die Information im System erfaßt wird oder
- auf den Zeitpunkt der Übergabe an den Monitor.

Mitentscheidend ist, wo die Zeitinformation bereitsteht. Wird die Zeitinformation innerhalb des Objektsystems bereitgestellt, so sind alle drei Zeitpunkte möglich. In dem Fall, wo außerhalb des Objektsystems, etwa in einem Hardware-Monitor, die Uhr implementiert ist, kann nur im Übergabepunkt die Zeitinformation der gemessenen Information zugeordnet werden.

Wie Information aus dem Objektsystem gewonnen wird, legt man bei der Plazierung eines Meßpunktes mit fest. Die extrahierbare Information und damit das Informationsvolumen wird durch den Meßpunkt festgelegt, aber auch durch den Abgriffsort: Meßfühler, die am Systembus angeschlossen sind, passen sich an die dort unterstützten Wortbreiten an, während Meßfühler an Hardware-Meßpunkten ein beliebiges Format erlauben. Dasselbe gilt für Meßpunkte, die durch Software-Prozeduren realisiert sind: beliebige Informationsformate lassen sich bei den Ereignisstrukturen bilden. Für eine spätere Auswertung ist es von Vorteil, wenn ein einheitliches Format verwendet wird, andererseits können verschiedene Meßfühler meßpunktabhängig unterschiedlichste Formate erzeugen und so ein optimales Format wählen. Im allgemeinen wird diese Flexibilität nur bei Meßpunkten gegeben sein, die in Form von Software-Prozeduren vorliegen.

Jede Information, die während einer Auftragsbearbeitung erzeugt wird, muß abgespeichert werden, wenn die Auswertung erst nach der Meßphase stattfindet. Zum Abspeichern gibt es folgende Ansätze:

- Separate externe Speichersysteme,
- Abspeichern in Speicherstellen eines zu messenden Programms,
- Speicherung im Systemspeicher des Objektsystems (unlimitiert),
- Reservierter, limitierter Bereich im Systemspeicher des Objektsystem,
- Abspeicherung [illegible]

- Übertragung über das Netz auf ein separates System zur Abspeicherung oder
- Übertragung über separates Monitornetz auf ein separates System zur Abspeicherung.

Externe Speicher verlangen, daß die Information bei der Erkennung durch den Monitor das Objektsystem verlassen [155, 200]. Die Beeinflussung des Objektsystems wird dann minimal. Grössere Fehler entstehen in den Fällen, bei denen Teile des Objektsystems zur Abspeicherung der Information benötigt werden. Doch darf dies nicht zu anderen Engpässen, etwa im Ein-/Ausgabesystem, und damit zu Verhaltensänderungen im Objektsystem führen.

Am einfachsten ist, wenn die Information im zu messenden System oder Programm selbst gespeichert wird. Alle erzeugten Informationen bleiben dem erzeugenden Prozeß zugeordnet, weiterhin kann dort der Speicherbedarf oftmals gut abgeschätzt werden. Terminiert ein Programm während der Meßphase, so muß die angesammelte Information an anderer Stelle gesichert werden, bevor der belegte Speicher an das System zurückgegeben werden kann. Ist die Meßphase kürzer als die Lebensdauer des zu messenden Prozesses, so muß ein Zugriff auf dessen Speicherbereich oder eine Transfermöglichkeit für die Meßdaten möglich sein [204].

Will man die Meßdaten nicht bei den Verursachern belassen, kann der Speicherbereich als Teil des Objektsystemspeichers angelegt sein: entweder als Datenbereich beliebiger Größe [137] oder als endlicher Bereich [64, 190]. Endliche Bereiche lassen sich einfach verwalten und der Kontrolle des Objektsystems entziehen, begrenzen jedoch die Meßdauer. Speicherbereiche ohne Limit verdrängen während einer Meßdauer andere Programme immer mehr aus dem Speicher und verändern damit das Objektsystem ganz massiv.

Ist das erwartete Meßdatenvolumen sehr groß, so ist nur ein Abspeichern auf Hintergrundspeicher wie Magnetplatte oder Magnetband sinnvoll. Dem Vorteil der großen Kapazität steht die Beeinflussung des Ein-/Ausgabesystems des Objektrechners gegenüber, das bei plattengestützten Betriebssystemen heutiger Rechenanlagen oft eine leistungsbegrenzende Komponente ist. Bei Objektsystemen ohne eigene Magnetplatte kann das Netz selbst mit einem darüber erreichbaren Fileserver zur Datenspeicherung verwendet werden [138]. Bei plattenlosen Systemen stellt nun der Netzzugang den Engpaß dar und wird somit noch zusätzlich belastet. Diese Nachteile lassen sich durch den Einsatz eines separaten Netzes verhindern, wenn ein weiterer Netzzugang am Objektsystem geschaffen wird [42, 82].

Neben diesen Fällen, in denen alle Meßdaten zusammen abgespeichert werden, existieren für einzelne Probleme eigene optimierte Ansätze. So kann beim Durchlaufen eines Netzpaketes durch einen Protokollstack an allen Meßpunkten die Zeit ermittelt und in aufeinanderfolgende Plätze im Paket selbst geschrieben werden: das zu beobachtende Paket sammelt quasi alle Informationen selbst und kann beim Empfang ausgewertet werden [159 (Timestamp-Option im Internet Protokoll IP), 35, 128]. Ein anderer Ansatz wird in [113] beim Einsatz auf einem Multiprozessorsystem angewandt, wo alle Meßdaten in einen gemeinsamen Speicherbereich kopiert werden und von dort durch einen Prozeß auf einem separaten Prozessor auf einen Hintergrundspeicher transferiert.

Eng mit dem Speicherproblem des Monitors ist das Problem der Informationskodierung verknüpft. Kodierte Daten belegen weniger Speicherplatz, benötigen jedoch Zeit für Kodierprozeduren oder geeignete Hardware-Schaltungen. Weiterhin entfällt durch Kodierung ein Teil der Information, der für die Auswertung somit nicht mehr zur Verfügung steht. Eine Kodierung während der Meßphase bedingt anschließend eine Dekodierung bei der Auswertung.

Kodierung ist eine Form der Vorverarbeitung. Eine weitere Form, das Datenvolumen während der Meßphase zu reduzieren, stellt die Filterung dar. Durch Auswahl von für diese Meßphase irrelevanten Informationen können Daten von aktivierten Meßpunkten ausgefiltert werden. Dies setzt auf der Hardware-Ebene schnelle Filterschaltungen voraus, da die erzeugten Daten vor einer möglichen Abspeicherung als unnötig erkannt werden müssen. Werden Monitorprozeduren in den Software-Schichten eingesetzt, so muß die Filterung über Abfragen erfolgen. Diese Abfragen sind immer in den Meßpunkten enthalten, was zu laufzeitintensiven Monitorprozeduren, d.h. Systembeeinflussungen, führt.

Ist eine Meßphase für einen Monitor einmal gestartet, so läuft diese meist ohne weitere Steuerungsmöglichkeit ab. So ist eine Veränderung der Meßpunkte, etwa eine Erzeugung, Aktivierung, Deaktivierung oder Entfernung, in den veröffentlichten Monitorrealisierungen nicht möglich.

Anschließend an die Messung, oder auch schon parallel zur Messung, findet die Interpretation der Informationen statt. Eine Aufgabe des Monitors ist, die Information so bereitzustellen, daß Weiterverarbeitung durch direkte oder statistische Auswertung und Anzeige möglich ist. Im Fall des Hardware-Monitors verzichtet man oft auf eine Datenspeicherung, die Visualisierung oder eine Weiterverarbeitung mit anschließender Visualisierung folgt direkt auf die Meßdatenerfassung [114, 135, 197].

Die Auswertung von Meßdaten baut auf einer Interpretation der Meßinformationen auf. Die bei der Meßpunktinstallation festgelegte Form muß nun dekodiert und ausgewertet werden. Eine Beschreibung der Formate mit einer formalen Sprache kann die automatische Installation der Meßpunkte unterstützen oder die Auswertung benutzerfreundlich machen. So wird in [82] die Sprache EDL zur Formatbeschreibung beschrieben. Eine relationale Datenbank mit spezieller Abfragesprache TQuel dient als Basis eines Konzeptes [190], bei dem über diese Abfragesprache die nötigen Meßpunkte initiiert und anschließend ausgewertet werden.

2.3.1 Hardware-Monitor

In [114] wird die Konfiguration einer Hardware-Messung angegeben als Kombination aus dem Monitor selbst und seinen Verbindungen zum Objektsystem:

- Meßfühler, die Hardware-Zustände eines Objektrechners erfassen (❶),
- Meßfühler, die an Hardware-Meßpunkten Rückschlüsse auf Objekt-Software erlauben (❷)

und

- Rückkopplungen (❸), um Meßergebnisse als Entscheidungshilfen im Objektsystem zu verwenden.

Die angegebene Rückkopplung erlaubt eine dynamische Adaptierung des Objektrechnerverhaltens, wird hier aber nicht weiter betrachtet.

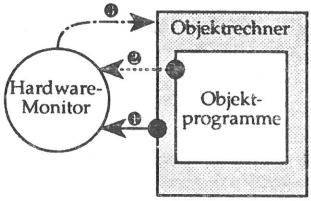


Bild 2.12: Konfiguration einer Hardware-Messung nach [114]

Meßfühler sind beim Hardware-Monitor in Hardware aufgebaut, um die Zustandsbedingungen beim Eintreten des auslösenden Ereignisses oder eine Zustandsfolge, die charakteristisch für eine Softwareaktion ist, zu erkennen. Dies verlangt, daß eine eindeutige Zustandsbedingung beim Durchlauf durch einen Meßpunkt vorliegt. Zugriffe auf spezielle Baugruppen oder Speicherstellen lassen sich so an ihrer am Systembus meßbaren Adresse erkennen. Dies gelingt bereits beim Beobachten von hochintegrierten Schaltungen nicht mehr, da oftmals der Meßpunkt nicht an den äußeren Anschlüssen liegt. Der Zugriff auf Programmvariablen kann bei bekannter Adresse beobachtet werden; nur im Fall ständig wechselnder Variablenadressen, wie es bei Stapelspeichern, Listen oder in Betriebssystemen mit virtuellem Speicher (Kachelspeicherprinzip) vorkommt, ist die Adreßzuordnung nicht mehr konstant. Durch Meßfühler, die die Adressierungsarten und die maschineninterne Darstellung der Variablenstrukturen kennen, wird auch in diesem Fall mit einigem Aufwand [82, 143] eine Beobachtung von Meßobjekten in sich wechselnden Speicherstellen möglich. Dies führt dazu, daß hochintegrierte Schaltungen nur durch weitere hochintegrierte Schaltungen beobachtbar sind, die einen Teil der Funktionalität des Objektsystems nachbilden können. Der Aufwand steigt dabei enorm [143], obwohl auch damit nicht alle Meßpunkte nachbildbar sind.

Geschwindigkeitsprobleme bei der Erfassung des Meßpunktes selbst treten kaum auf. Die anschließende Vorverarbeitung ist zeitkritisch, wenn Teile der Auswertung in diesem Schritt zur Datenreduktion durchgeführt werden müssen: so ist die Berechnung einer Rate bei einem minimalen Ereignisabstandes im Monitor von [143] nicht durchführbar, weshalb die einzelnen Zählerwerte als Teil des Ergebnisses für die spätere Divisionsrechnung bereitgestellt werden müssen.

Meist handelt es sich bei den Auswertungsschaltungen um Zähler (Summen- oder Phasenzähler), Vergleicher oder Paarbilder, und die Auswertung entfällt, so daß die Meßwerte unmittelbar auf einer Anzeige präsentiert werden, z.B. in Form von Lichtsignalen oder Balkenanzeigen.

Das Problem der großen meßbaren Datenmenge verlangt eine Reduktion auf Hardware-Ebene und kann dem Aktivieren/Deaktivieren eines Meßfühlers entsprechen. Differenziertere Meßfühler lassen eine Datenreduktion durch Maskierung einzelner Meßpunkte zu: entweder wird die Meßpunkterkennung nicht durchgeführt oder die erkannten Meßpunkte werden nicht weitergemeldet.

Den Vorteilen eines Hardware-Monitor:

- gleichzeitiges Beobachten an vielen Meßpunkten,
- keine Systemveränderungen,
- hohe zeitliche Auflösung,
- Meßpunkte auf unterster Hardware-Ebene möglich,

- Beobachtung in instabilen Systemzuständen (Systemstart, Systemabsturz),
- Meßpunkte in für die Software nicht zugänglichen Stellen (DMA, E/A-Kanäle),

stehen viele Nachteile gegenüber:

- Information nur durch Beobachten extrahierbar,
- Anschluß der Meßfühler kritisch,
- Meßanordnung schlecht änderbar (nicht während des Betriebs) und
- Verständnislücke zwischen Hardware-Zustand, Software-Zustand und Objektsystemverhalten nur durch detaillierte Systemkenntnis zu schließen.

Gerade der letzte Punkt macht deutlich, daß ein Hardware-Monitor nicht das optimale Werkzeug zum Erwerb von Systemkenntnissen ist: viele Kenntnisse über das interne Objektsystemverhalten werden im voraus verlangt. Trotzdem wird er eingesetzt, um die erwarteten Abläufe zu überprüfen [155, 200]. Eine spezielle Form ist der Netz-Tester oder Protokollanalysator, der zur Erkennung von Nachrichten im Lokalen Netz (engl. Local Area Network, LAN) oder an einer anderen Netzschnittstelle optimiert wurde.

2.3.2 Software-Monitor

Im Gegensatz zum Hardware-Monitor ist kein zusätzlicher Aufwand in Form von Baugruppen, Meßfühlern oder Spezialschaltungen nötig. Alle Monitoraktivitäten laufen Software-gesteuert ab und erreichen deshalb nur Zustände von Programmen.

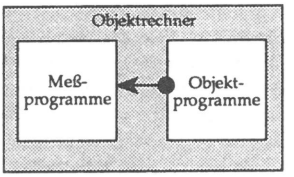


Bild 2.13: Konfiguration einer Software-Messung nach [114]

Wie auch beim Hardware-Monitor kann der Anstoß zu einem Meßpunkt von intern oder extern kommen. Extern bedeutet hier von einem weiteren (periodischen) Prozeß innerhalb des Objektsystems, der damit das Sampling anstößt, während das Durchlaufen eines im Programmablauf auftretenden Meßpunktes ein Tracing erlaubt.

Da Software, d.h. einige spezielle additive Befehle, am Meßpunkt eingefügt sind, kann nur auf die dort sichtbaren Variablen zugegriffen werden: der Sichtbarkeitsbereich grenzt die möglichen Stellen eines Meßpunktes ein. Der genaue Ort muß wiederum fallspezifisch festgelegt werden. Entscheidend ist, daß hier das Durchlaufen eines Meßpunktes (der Monitorprozedur) Zeit beansprucht. Alle Schritte von Filterung über Vorverarbeitung und Zwischenspeicherung sind während dieser Zeit durchzuführen. Neben dieser unerwünschten zeitlichen Beeinflussung kann die Monitorprozedur Nebeneffekte haben, die das Systemverhalten weiter beeinträchtigen: Zugriffe auf Va-

riablen verändern die Historie in einem Datenschnellspeicher (engl. Data Cache), die zusätzlichen Befehle laden neue Befehle in die Abarbeitungskette (engl. Command Pipeline) und verdrängen andere Befehle aus den Befehlsschnellspeichern (engl. Command Caches), in Betriebssystemen mit Kachelspeicher kann dies Einfluß auf die aktuell sich im Speicher befindlichen Seiten und somit Seitenwechsel zur Folge haben.

Wie in Kapitel 2.1.2 bei der Definition eines Meßpunktes beschrieben, muß der Erfassungszeitpunkt diesem Meßpunkt zugeordnet werden. Hier muß auf systeminterne Uhren zugegriffen werden, was in üblichen Betriebssystemen viel Zeit kosten kann: [142] gibt die benötigte Zeit zum Lesen der Uhrzeit in BSD-UNIX mit typ. 660 - 840 µs an. Die Gesamtzeit in einer Monitorprozedur hängt dann noch von der Meßpunktfilterung, der Vorverarbeitung und der Speicherung ab.

Art	Laufzeit (µs)	≈STORE-Befehle
inaktiver Sensor	15	2
ausgefilterter Sensor	90...165	45...80
aktiver Sensor	600...1400	85...200

Tabelle 3: Laufzeiten bei Monitorprozeduren nach [190]

In Tabelle 3 sind typische Laufzeitwerte einer Mikroprogramm-kodierten Monitorprozedur aufgeführt, Versionen mit Prozeduraufrufen kosteten dagegen 1850...2330 µs Zeit. Bei diesen Laufzeiten ergibt sich eine minimale zeitliche Granularität, die bestenfalls bis zur Prozedurebene reicht; kürzere Vorgänge können nicht mehr zeitlich unterschieden werden. Noch gröbere Granularitäten werden beim Sampling in Kauf genommen, um die Systembeeinflussung klein zu halten. [211] berichtet von einem Software-Monitor, bei dem nur alle 5...100 ms das System angehalten und dessen Zustand analysiert wird.

Da die innerhalb eines Systems verfügbaren Uhren schlechte zeitliche Auflösungen (UNIX-Systeme 20 ms, SUN-Systeme 10 ms) haben, werden meist viele Meßdurchläufe gestartet und durch Mittelwertbildung die einzelnen Laufzeiten zwischen zwei Ereignissen gebildet. Die vorhandene Granularität der Zeitgeber ist dabei meist größer als die zu messenden Zeitintervalle, so daß die Mittelwertbildung erst ab größeren Stichprobenanzahlen gültig wird. [34] gibt eine Abschätzung an, wieviele Intervalle mindestens gemessen werden müssen, um signifikante Nachkommastellen (bei vorgegebener Uhrenauflösung) angeben zu können (vergl. Tabelle 4).

zu messende Intervalldauer	2 sign. Nachkommastellen	3 sign. Nachkommastellen
10 µs	800.000	80.000.000
100 µs	80.000	8.000.000
1 ms	7.000	700.000
10 ms	400	40.00

Tabelle 4: Anzahl notwendiger Meßintervalle bei zeitlicher Auflösung Δ=20 ms und Vertrauensintervall α=0.95 nach [34]

Eine Verbesserung der Uhrenlesezeit und auch der zeitlichen Auflösung um den Faktor 30 bringt der Einbau von Zeitgebern, wie er in [34] vorgeschlagen und in [128] verwendet wurde.

Der Speicher stellt beim Software-Monitor ein weiteres Problem dar. In realisierten Software-Monitoren wurde der Speicher als:

- Informationspool [211],
- Objektprogramm-interner Speicher [138],
- Objektsystem-interner Speicher [113, 137, 190] oder
- Objektsystem-interner Ringpuffer [201]

ausgeführt. So wurden in [137] zwischen 1 und 36 MBytes (typ. 16 MBytes) Speicher vom gesamten Objektsystem (Betriebssystemspeicher 64 MBytes) abgezogen. Beim Einsatz kleinerer Speicherbereiche wie in [201, 83] mußte zur Vermeidung von Überschreibungen der Puffer periodisch ausgelesen werden, weshalb die Messung für die Dauer des Datentransfers unterbrochen wird.

Software-Monitor	Fehler	Bemerkung	Literatur
in Firmware	1...2,7 %	je Meßfühler im Objektsystemspeicher ein Häufigkeits- und Intervalldauerzähler	[64]
in Mikroprogramm	typ. 1 %	nicht alle Meßfühler aktiv, Speicher im Objektsystem	[190]
in Mikroprogramm	< 10 %	Speicherung in Objektsystempuffer	[137]
	0,3...1,8 %	Befehlszähler (PC)-Sampling mit Zählern im Objektsystemspeicher	[134]
in Assembler	50...200 %	Zähler für jeden Kodeblock in Programmspeicher	[204]
in Hochsprache	typ. 10 %	konkurrierender Zugriff mehrerer Prozesse auf zentralen Puffer	[113]
	0,1...10 %	Speicherung in jeweiligem Objektsystemspeicher	[166]
		PC-Sampling alle 5...100 ms, Speicherung in verteilten Puffern	[211]

Tabelle 5: Laufzeitveränderungen bei Software-Monitoren

Die durch den Einsatz eines Software-Monitors erzeugte Laufzeitveränderung wird allgemein vernachlässigt, wenn diese unter 5% liegt [135, 176], vergleiche Tabelle 5.

Wie auch bei Hardware-Monitoren stehen sich hier den Vorteilen:

- Programme lassen sich einfach um Monitorprozeduren erweitern,
- Zugriff auf alle Softwarestrukturen jederzeit möglich,
- dynamische Vorgänge (virtuelle Speicherverwaltung, dynamische Datenstrukturen, Rekursion) sind leicht zu messen

auch Nachteile gegenüber:

- große Systembeeinflussungen,
- nur für Messungen in Programmen geeignet,
- Phasen ohne Programmaktivität schlecht meßbar
- zeitliche Auflösung abhängig vom Objektsystem,
- keine Messungen an mehreren Stellen gleichzeitig möglich und
- Hardware-Ereignisse ohne zugeordnete Software-Aktion nicht erkennbar.

2.3.3 Hybrid-Monitor

Die Idee beim Hybrid-Monitor ist es, alle Vorteile der beiden Konzepte (Software- und Hardware-Monitor) zu vereinen, ohne die Nachteile zu übernehmen (Bild 2.14).

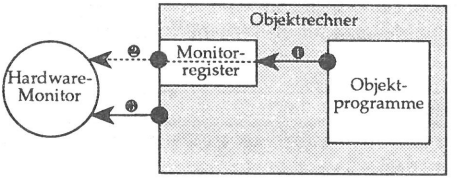


Bild 2.14: Konfiguration für Messung mit Hybrid-Monitor

Vom Konzept des Hardware-Monitors versucht man, die geringe Systembeeinflussung zu erhalten und ergänzt dieses durch die Fähigkeit des Software-Monitors, Messungen in den reinen Software-Schichten zu machen: die Monitorprozeduren des Software-Monitors legen nun ihre Informationen nicht mehr in einem Speicher ab, sondern stellen diese über ein Monitorregister (❶) einem externen Hardware-Monitor zur Übernahme (❷) bereit. Die Zuordnung eines Zeitwertes kann nun entweder im Objektrechner oder extern im Hardware-Monitor erfolgen. Zusätzlich bleiben parallel dazu die vielfältigen Meßmöglichkeiten des Hardware-Monitors (❸) erhalten.

Die Vorteile dieses Konzepts kommen zum einen durch die parallele Nutzung von Hardware-nahen und Software-orientierten Messungen zum tragen, aber auch die massiven Systembeeinträchtigungen durch die Monitorprozeduren verringern sich. Das Monitorregister wird an den Objektrechner angepaßt und stellt sich für die Monitorprozedur als Ein-/Ausgabestelle oder noch besser als Speicherzelle dar. Die prinzipielle Arbeitsweise zeigt Bild 2.15.

Die im Monitorregister implementierte Logik sorgt für eine Entkopplung der Vorgänge zwischen Hardware-Monitor und Objektrechner, so daß dieser, ohne auf die Übernahme oder Abspeicherung zu warten, sofort weiterlaufen kann. So werden in [142] als Laufzeit für eine Wertzuweisung an ein Monitorregister (im folgenden *Meßstatement* genannt) typ. 3...8 µs angegeben, was etwa zwei bis vier Maschinenbefehlen entspricht. Diese kurze Laufzeit drückt die Systemverlangsamung

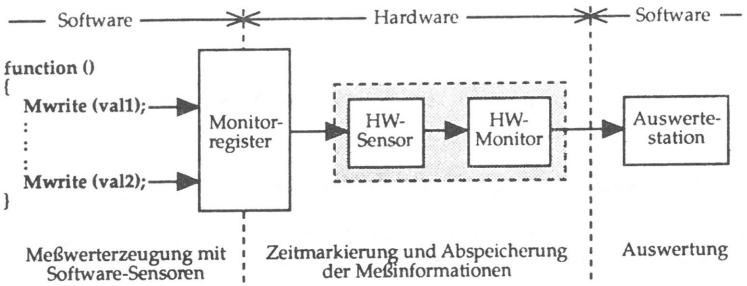


Bild 2.15: Monitorregister als Bindeglied zwischen Objektsystem und Monitor nach [52]

auf Werte deutlich unter 0.1% [68, 208], so daß in vielen Fällen deshalb kein Anlaß mehr zur Entwicklung von Hardware-Meßfühlern besteht. Tabelle 6 gibt einen Überblick über erzielte Werte.

HW-Meßfühler	Monitorregister	Dauer für Registerzugriff	zeitliche Auflösung	Literatur
nein	8bit parallel, E/A-Schnittstelle		8 µs	[42]
nein	2*16bit parallel, Speicherzelle	4 µs	1...100 µs	[68]
ja	8bit parallel, E/A-Schnittstelle	5 µs	100 ns	[82]
nein	16bit parallel, Speicherzelle	3...8 µs	1 µs	[142]
ja	32bit, Speicherzelle	1 µs	1 µs	[183]

Tabelle 6: Charakteristische Größen einiger Hybrid-Monitore

Bei all diesen Hybrid-Monitoren wurde das Speicherproblem in ähnlicher Weise gelöst: die von den Meßfühlern erkannten Daten werden in einem Meßteil extern zwischengespeichert und die Zuordnung des Zeitstempels erfolgt innerhalb des Meßteils.

3 Meßproblematik in verteilten Systemen

3.1 Leistungsmessung an einem System

Der einfachere Fall ist, daß nur an einem System in einem Netz Messungen durchgeführt werden müssen. Hierzu reichen die in Kapitel 2 beschriebenen Konzepte völlig aus.

Lokal erfolgt die Erzeugung und Eintragung der Meßinformationen in eine Ereignisspur. Diese wird entweder lokal oder in einem entfernten System analysiert.

3.2 Leistungsmessung an verteilten Systemen

Aufwendiger ist es, wenn zusätzlich Messungen netzweit erfolgen sollen. Jetzt erzeugen alle an der Messung beteiligten Objektsysteme Informationen in dezentraler Weise. Aktionen, die auf einem Objektsystem angestoßen werden, können Bearbeitungsschritte auf einem oder vielen anderen Objektsystemen initiieren. Die dabei zu messenden Größen sind nur bei einer gleichzeitigen globalen Sicht über alle beteiligten Systeme zu erhalten.

Wird als Meßkonzept zur verteilten Leistungsmessung ein Monitor eingesetzt, so entstehen mehrere Ereignisspuren der räumlich verteilten Objektsysteme, die in der Auswertephase zu analysieren sind.

3.2.1 Messung an einem Meßobjekt

Dieser Spezialfall entspricht dem aus Kapitel 3.1, so daß hierzu ein verteiltes Meßsystem nicht nötig ist. Ein Konzept zur Messung in verteilten Systemen muß diesen Fall jedoch abdecken können.

Durch zeitlich aufeinanderfolgende Messungen an unterschiedlichen Systemen lassen sich auch Leistungsergebnisse in verteilten Systemen gewinnen, jedoch eine direkte Einsicht in die zeitlichen Abhängigkeitszusammenhänge ist nicht möglich.

3.2.2 Messung an mehreren Meßobjekten

Eine Messung an mehreren Systemen wird erst durch die gleichzeitige Informationserzeugung, und -erfassung in allen beteiligten Objektsystemen ermöglicht.

Wie aus Bild 3.1 ersichtlich, kann der Einfluß weiterer Objektsysteme nur dann auf die Bearbeitungszeit eines Auftrags am System *B* erkannt werden, wenn neben dem eigentlichen Objektsystem *A* noch mindestens das System *B* gleichzeitig meßtechnisch erfaßt wird. Die Verlängerung der Bearbeitungszeit T_2 im Vergleich zu T_1 kommt durch wichtigere Aufträge der weiteren Objektsysteme *C* und *D* zustande. Im Falle der alleinigen Betrachtung von Objektsystem *A* kann diese zusätzliche Last in *B* nicht angegeben werden. Das Objektsystem *D* wird bei b) nicht gemessen, dessen Einfluß auf das Meßobjekt *B* läßt sich aber angeben.

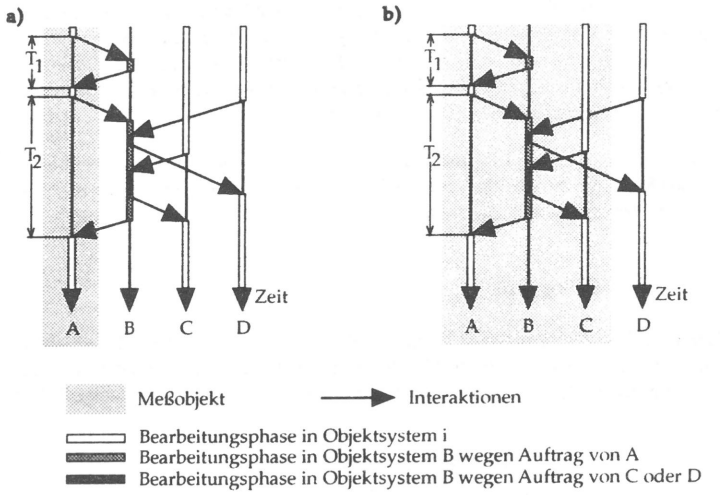


Bild 3.1: Sichtweite bei a) einem und b) mehreren Objektsystemen

Können die Meßpunkte in verteilten Objektsystemen sowie deren Auftrittszeitpunkte erkannt werden, so besteht die Möglichkeit, die Einwegverzögerung direkt zu messen, entsprechend Bild 3.2b).

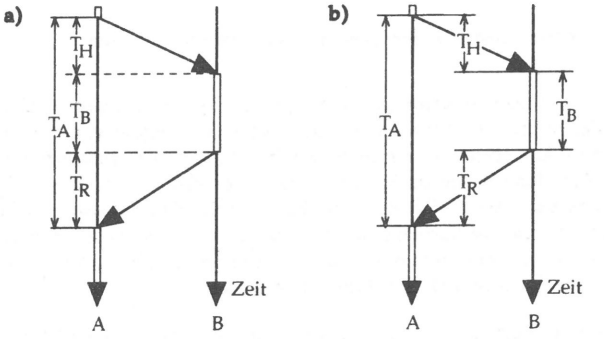


Bild 3.2: Einwegverzögerung: a) Laufzeithalbierung [146, 158], b) direkte Messung

Die fehlerträchtige Halbierung der Auftragsbearbeitungszeit T_A aufgrund der nicht einzeln meßbaren Hinlaufzeit T_H, der geschätzten Bearbeitungszeit T_B und der Rücklaufzeit T_R entfällt. Darüberhinaus können dann auch einzelne Einwegverzögerungen T_H und T_R gemessen werden, während beim Halbieren der Laufzeiten immer die Mittelwerte der Laufzeiten genommen werden.

Eine direkte Messung der Einwegverzögerung (hier T_H und T_R) verlangt, daß die begrenzenden Zeitpunkte durch eine Uhr, basierend auf der physikalischen Zeit, zeitlich erfaßt sind. Eine globale Uhr scheidet meist aufgrund der örtlich verteilten Lage der Objektsysteme aus, so daß ein Ersatz für diese globale Uhr in Form einer zentralen Zeitbasis und einem dazupassenden Zeitverteilungsmechanismus oder verteilten Uhren mit notwendiger Synchronisation erforderlich wird.

Die Leistungsmeßkonzepte von ISO und Internet erlauben nur lokale Messungen. Die alternativen Konzepte eines Monitors können für verteilte Systeme erweitert werden, wobei auch hier das Synchronisationsproblem gelöst werden muß.

3.3 Synchronisation

Das Problem der Zeitsynchronisation besteht in allen verteilten Systemen. So existieren in vielen Protokollarchitekturen spezielle Protokolle, die durch Austausch eine einheitliche Zeitbasis unterstützen. Eine Einteilung der grundsätzlichen Synchronisationsmöglichkeiten zeigt Bild 3.3.

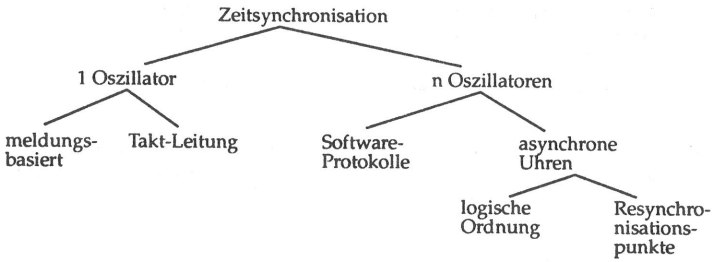


Bild 3.3: Klassifizierung der Synchronisationsmöglichkeiten

Daß es auch ohne explizite Synchronisation geht, hat [121] gezeigt. Dabei wird jedoch eine logische Ordnung der Ereignisse, zumindest abschnittsweise, vorausgesetzt. Diese partielle Ordnung erlaubt nur relative Aussagen wie „ereignete sich bevor", kann aber zusammen mit dem Austauschen der verteilten Uhrenstände zur Synchronisation (im Sinne von zeitlicher globaler Ereignisordnung) verwendet werden. Auf diesen logischen Uhren aufbauend kann bei Berücksichtigung der Laufzeit der Austauschmeldungen eine Abschätzung für die mögliche Uhrengenauigkeit angegeben werden, wobei die Zeitpunkte beim Eintreffen der Austauschmeldungen als neue synchrone Punkte auf der Zeitachse dienen (Resynchronisationspunkte).

Nachteilig bei diesem Verfahren ist, daß mit jedem Meldungsempfang die lokale Uhr einen Sprung machen kann, d.h. die Monotonie verletzt wird. Dasselbe gilt auch für den in [148] beschriebenen Uhrzeitsender, bei dem an einer zentralen Stelle eine Registrierung für periodische Zeitübertragungen erfolgen kann. Die jeweils aktuelle Zeit wird dann über eine Schicht 4-Verbindung (ISO TP4) verteilt und zur Korrektur der lokalen Zeit verwandt.

Einen für große Netze geeigneten Weg geht [140] mit dem Network Time Protocol (NTP), bei dem wenige hochgenaue Zeitbasen (Primary Servers) ausfallsicher eine sehr genaue Zeit (mit Berücksichtigung der Zeitzonen) anbieten. Von diesen Zeitbasen wird die aktuelle Zeit verteilt, wobei

die Netzeinflüsse über Mittelungen der gemessenen Laufzeiten mitberücksichtigt werden. In [139, 141] wird die mit dem Protokoll NTP erzielbare Zeitgenauigkeit angegeben, vergl. Tabelle 7.

T [in Sekunden]	P{Fehler ≥ T} ohne Filterung	P{Fehler ≥ T} mit Filterung
0.001	0,98	0,93
0.01	0,53	0,079
0.1	0,05	≈ 0
1	0.002	≈ 0
10	0,0007	≈ 0

Tabelle 7: Zeitabweichungswahrscheinlichkeit ohne bzw. mit Filterung nach [141]

Die mit NTP erzielbare Abweichung der Zeit in den Primary Servern ist entsprechend Tabelle 7 sehr gering. [141] gibt an, daß im Mittel der Zeitfehler nur wenige Millisekunden betrug, während die maximale beobachtete Zeitabweichung nicht größer als 50 Millisekunden war.

Außer dem NTP, das in [196] zur Systemsynchronisation während der Authentifikation dient, existieren weitere Verfahren, die meldungsgestützt verteilte Uhren synchronisieren. So gibt [75] 20...50 ms als erreichbare Genauigkeit des von [66] entwickelten Verfahrens in einer LAN-Umgebung an.

Allen bisher beschriebenen Synchronisationsverfahren ist gemeinsam, daß Meldungen zur Synchronisation eingesetzt werden. Üblicherweise wird dazu das ohnehin vorhandene Netz verwendet. Reicht die mit diesen Verfahren erreichbare zeitliche Genauigkeit nicht aus oder kann man die dabei generierte Netzlast nicht erlauben, so wird der Einsatz von Hardware-Erweiterungen nötig.

Die Möglichkeiten, wie die Zeitinformation bereitgestellt und wo demzufolge die dezentral erfaßten Informationen zeitlich markiert werden, lassen sich mit dem von [1] vorgeschlagenen Schema gut darstellen (Bild 3.4). In dieser abgekürzten Schreibweise enthält jedes System folgende Komponenten:

O	Oszillator,
C	Uhr (engl. Counter),
M	Monitor,
S	Speicher und
H	Objektsystem (engl. Host),

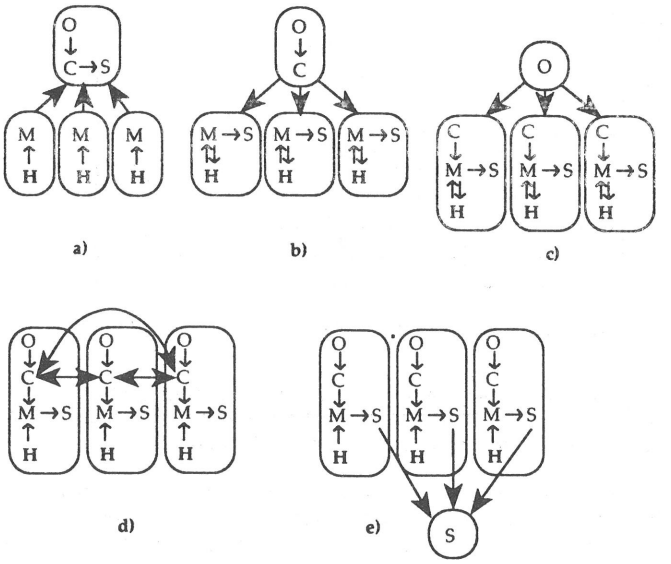


Bild 3.4: Zeitbereitstellung, Zeitmarkierung und Speicherung

wobei alle Komponenten innerhalb einer Umrandung als eine selbständige Einheit betrachtet werden. Das Objektsystem (H) wird vereinfachend als Teil der verteilten Monitorkomponenten angegeben. Die gerichteten Pfeile beschreiben den Informationsfluß. Die doppeltgerichteten Pfeile zwischen Objektsystem (H) und Monitor (M) zeigen an, daß sowohl Tracing als auch Sampling möglich ist.

In Variante a) werden die Informationen von verteilten Monitoren erfaßt und an eine zentrale Komponente zur Speicherung gesandt [76]. Diese Variante braucht keine Zeitsynchronisation, da die Informationen von den verteilten Monitoren zum speichernden System transferiert werden und dort entsprechend der Ankunftszeit markiert und abgelegt werden. In Variante b) wird die Zeitinformation verteilt, so daß die Informationen mit dem Zeitstempel in den Monitoren gespeichert werden [148], während Variante c) nur eine Zählinformation (Taktpulse) an die verteilten Monitore liefert [42, 70]. Die von [66, 121, 140] vorgeschlagenen Synchronisationsverfahren entsprechen Variante d), während Variante e) gänzlich ohne Zeitsynchronisation und Informationstransfer auskommt, jedoch die Speicherinhalte zu Synchronisationszwecken analysieren muß. Im schlechtesten Fall wie in [113] kann die richtige zeitliche Reihenfolge der Ereignisse nicht mehr rekonstruiert werden.

Beim Verteilen der Zeit gibt es zwei Möglichkeiten zur Korrektur der Systemzeit, falls die verteilten Uhren nicht synchron laufen:

- Nachstellen (Stetigkeitsverletzung) oder
- Nachziehen (Verlangsamen oder Beschleunigen der Uhrenfrequenz).

Am einfachsten ist es, im System die Uhr einfach auf den neuen Zeitwert zu stellen ohne zu prüfen, ob der alte Zeitwert abweichend, d.h. größer oder kleiner, war. Mit größerem Aufwand (nur durch eine geeignete Hardware in Form einer Phase Locked Loop, PLL) kann die lokale Uhr allmählich auf den Sollwert gezogen werden. Eigenschaften der Synchronisationsverfahren listet Tabelle 8 auf.

Verfahren	Zeitverlauf	Zeitzuordnungs-variante nach Bild 3.4
meldungsbasiert	Zeitsprünge (vor- und rückwärts)	b) und d)
Takt-Leitung	stetig	c)
Software-Protokolle	stetig Zeitsprünge (vorwärts) Zeitsprünge (vor- und rückwärts)	d)
logische Ordnung	Zeitsprünge (vorwärts)	e)
Resynchronisationspunkte	stetig	e)

Tabelle 8: Zeitverläufe bei unterschiedlichen Synchronisationsverfahren

Problematisch bei den Verfahren, die die Zeitinformation verteilen, ist, daß im Falle von Verlusten von Taktpulsen oder Meldungen die Zeit in den verteilten Monitoren unterschiedlich weiterlaufen kann. Dies verlangt fehlertolerante und damit aufwendige Übertragungsverfahren [82, 148], damit eine durchgeführte Messung nicht umsonst war.

3.4 Speicherung der Meßinformation

In allen Modellen eines verteilten Monitors gemäß Bild 3.4 sind Speicher enthalten. Wird auf diese Speicher verzichtet, so muß die Auswertung schritthaltend erfolgen, da die aktuell erfaßten Informationen sofort wieder verloren gehen.

Sind Speicher vorgesehen, so können diese Speicher verteilt oder zentralistisch realisiert sein. Modell a) aus Bild 3.4 besitzt einen zentralen Speicher, zu dem alle Informationen während der Messung transferiert werden müssen. Andererseits dient ein dezentraler Speicher in den Modellen b) bis e) aus Bild 3.4 als temporärer Zwischenspeicher, aus dem anschließend an die Messung die Daten zur netzweiten Auswertung an eine zentrale Stelle übertragen werden müssen. Als Übertragungsweg stehen die in Bild 3.5 gezeigten Möglichkeiten zur Verfügung.

Ausgehend vom logischen Modell eines verteilten Meßsystems gibt es drei Möglichkeiten:

- Monitor als Teil innerhalb des Objektsystems (Bild 3.5a),
- Monitor als selbständige Einheit mit eigenem Netzzugang auf das vorhandene Netz *N* (vergl. Bild 3.5b) oder
- Monitor als selbständige Einheit mit eigenem Netzzugang auf ein spezielles Monitornetz *MN* (vergl. Bild 3.5c).

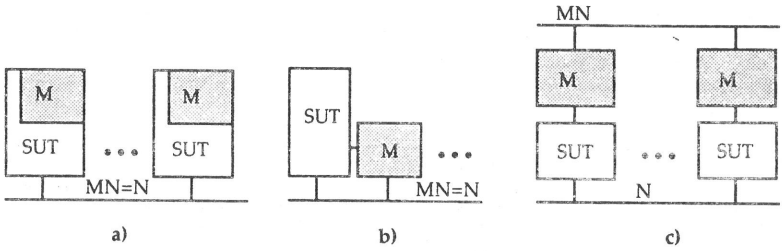


Bild 3.5: Verbindung der Monitorkomponenten (MN=Monitornetz, N=vorhandenes Netz)

Im ersten Fall nutzt der Monitor alle Ressourcen des Objektsystems wie Speicher und Netzzugang, und die Beeinträchtigung dieses Systems ist demzufolge groß. Weniger Einfluß auf das Objektsystem entsteht, wenn der Monitor als eigenständige Einheit mit einem eigenen Netzzugang versehen ist. Über diesen Netzzugang lassen sich alle Meßinformationen zu einer Station, die eine netzweite Ereignisspur bildet, übertragen. Dasselbe kann mit dem in Bild 3.5c gezeigten speziellen Monitornetz erreicht werden, nur gibt es hier keinerlei Beeinträchtigung des zu messenden Objektsystems und des vorhandenen Netzes. Tabelle 9 stellt die Vorteile den Nachteilen der Verbindungsschema gegenüber.

Verbindungsschema	Vorteile	Nachteile
Monitor integriert in Objektsystem	nur 1 Netz notwendig, einfach	Einfluß auf Objektsystem, Einfluß auf Netz
Monitor verwendet vorhandenes Netz	kein Einfluß auf Objektsystem, nur 1 Netz notwendig	aufwendig, Einfluß auf Netz
spezielles Monitornetz	kein Einfluß auf Objektsystem, kein Einfluß auf Netz	aufwendig, 2 Netze notwendig

Tabelle 9: Vor- und Nachteile der Monitorverbindungsschema

Um die Systembeeinflussung minimal zu halten, ist es, abhängig vom Monitorprinzip, entweder besser die Informationen direkt abzuspeichern, was einen großen Speicherbedarf zur Folge hat, oder die gewünschte Information in eine kompaktere Form zu kodieren und dann abzuspeichern. Der zur Kodierung benötigte Rechenaufwand wird meist akzeptiert, um möglichst viele Daten im Speicher sammeln zu können.

Bevor die eigentliche Messung durchgeführt werden kann, wird die Art und Menge der zu erfassenden Information in den Meßpunkten festgelegt. Damit wird gleichzeitig die interne Darstellung, Kodierung und Form der gespeicherten Daten definiert. Üblicherweise sind in einem Event Record neben der kodierten Information selbst zusätzliche Felder für

- Zeitstempel,
- Validierung der Zeitstempel (Überlauf, Stetigkeit),
- Systemkennung und

- interner Zustand des Monitors (Speicherzustand, Ereignisverluste)

enthalten.

Die Systemkennung erlaubt es, daß gleichartige Ereignisse auf unterschiedlichen Objektsystemen in der Ereignisspur unterschieden werden können. Diese Kennung muß systemweit eindeutig sein, d.h. muß für die Dauer einer Messung konfigurierbar sein.

Eine einfache Kennung des auslösenden Ereignisses wird in [42] verwendet. [82] definiert eine Beschreibungssprache zur benutzerfreundlichen Beschreibung der kodierten Informationen in den Event Records. Damit lassen sich Einträge in den Ereignisspuren beschreiben und bei der Auswertung über Namen selektieren und ansprechen. Die Erkennung, um welches Ereignis es sich beim vorliegenden Eintrag in der Ereignisspur handelt, wird ebenfalls über eine eindeutige Kennung an spezieller Stelle des Event Records gemacht. Weitere Felder können in ihrer Länge festgelegt, benannt und mit Wertebereichen angegeben werden.

Obwohl die Auswertung in [42] und [82] auf diesen Kennungen basiert, wird in keinem System die Beschreibung zur automatischen Generierung der Event Records verwendet oder diese Beschreibung automatisch erstellt.

3.5 Auswertung

Während des Auswertungsschritts dienen die Meßdaten zur

- Fehlersuche
- Einzelauswertung oder
- statistischen Auswertung.

3.5.1 Fehlersuche

Ist die Beschreibung der Meßdaten je Ereignis bekannt, so kann durch Interpretation der einzelnen Felder der Event Records jede Information dekodiert und angezeigt werden. Über die zugeordneten Zeitstempel können so vom Benutzer das Objektsystemverhalten nachvollzogen und eventuelle Verhaltensfehler erkannt werden. Dies setzt bereits eine fehlerfreie Kodierung der Informationen in den Event Records in der Ereignisspur voraus.

3.5.2 Direkte Auswertung

Wird eine einfache Ereigniskennung wie in [42] verwendet, so kann eine direkte Auswertung nach der Dekodierung, basierend auf der Datenbeschreibung, erfolgen. Dabei wird jeder Event Record einzeln ausgewertet und dem Benutzer angezeigt. Über einfache Darstellungen zur Visualisierung wie dem Gantt-Diagramm sind schnell Einblicke über das Ablaufgeschehen erzielbar.

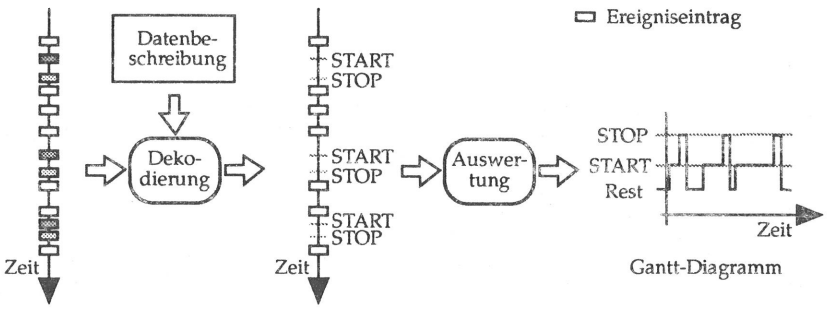


Bild 3.6 Direkte Auswertung mit Gantt-Diagramm

Das Objektsystemverhalten kann direkt aus dem Diagramm abgelesen werden, die Zustandswechsel (in Bild 3.6 zwischen START, STOP und Rest) sind durch die Event Records ausgelöst, die in der Ereignisspur gefunden wurden. Die zeitliche Dauer entspricht dem Abstand zwischen zwei Ereignissen im Diagramm.

Die direkte Auswertung ist einfach und kann deshalb auch auf Daten angewandt werden, die nicht erst abgespeichert wurden. Bleibt die Anzahl von unterschiedlichen Ereignissen klein ([72] verwendet bis zu 17, [82] verwendet nur 3), dann wird eine grafische Darstellung wie in Bild 3.6 in Realzeit während der laufenden Messung möglich.

3.5.3 Statistische Auswertung

Zwei Arten von Auswertungen sind möglich:

- Statistik über ein Objektsystem und
- netzweite Statistik.

Der ersten Auswertungsart genügen die Informationen, die für dieses Objektsystem relevant sind. Liegen diese Daten in einer auf dieses Objektsystem bezogenen Ereignisspur vor, so ist auch dezentral eine sofortige oder spätere Auswertung möglich. Vorteilhaft ist, daß nicht alle im Netz gemessenen Daten bei der Auswertung mitberücksichtigt werden müssen und ein Transfer der Ereignisspur an eine zentrale Auswertestation entfallen kann. Dafür ist an dezentraler Stelle die erforderliche Rechenleistung vorzusehen. Weiterhin kann dies bedeuten, daß die Beschreibung der Ereigniskodierung erst dieser lokalen Komponente bekanntzumachen ist.

Stationsübergreifende Messungen sind nur über die netzweite Sicht über eine globale Ereignisspur möglich. Diese wird an zentraler Stelle aus den einzelnen Ereignisspuren gebildet, so daß nur hier diese Auswertungsart erfolgen kann. Dies verlangt eine leistungsfähige Rechenanlage, da diese die globale Ereignisspur verwalten und andererseits die zur Auswertung erforderliche Rechenleistung stellen muß. Dafür läßt sich in den verteilten Komponenten Aufwand einsparen.

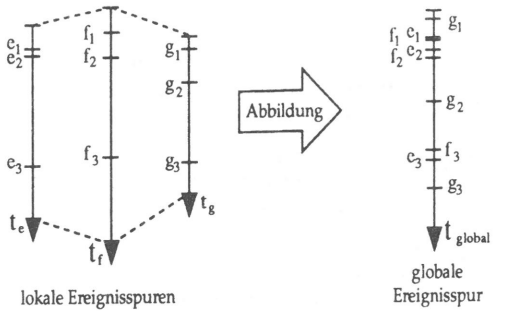


Bild 3.7: Bildung einer globalen Ereignisspur

Je nach dem eingesetzten Synchronisationsverfahren beinhaltet dieser Abbildungsprozeß eine einfache Sortierung, gesteuert über die Zeitstempel, oder einen komplexen Rechenschritt mit Dateninterpretation, Korrektur der Zeitstempel und nachfolgender Sortierung. Erst anschließend erfolgt die statistische Auswertung der globalen Ereignisspur.

Statistische Auswertungen während einer laufenden Messungen sind nur für die einfachsten Größen wie Ereigniszähler oder Intervallängenmessung in Realzeit möglich. So ist selbst im Hardware-Monitor von [143] die Berechnung von Raten nicht zeitsynchron durchführbar. Sind Quantile oder zentrale Momente zu berechnen, so muß der Mittelwert zur Berechnung verfügbar sein. Auch bei der Histogrammbildung ist die Kenntnis des minimalen und maximalen aufgetretenen Wertes für eine Anzeige nötig, um die Klassenbreite sinnvoll zu wählen. Verwendet man Schätzwerte [108] als Basis zur Berechnung, so ist auch eine Online-Auswertung für die Größen möglich, wie sie in den Empfehlungen der CCITT (wo neben den Mittelwerten auch die Werte der 95%-Quantile einzuhalten sind) verlangt werden.

Bei gespeicherten Event Records in der globalen Ereignisspur können alle mathematischen Algorithmen eingesetzt werden. Weiterhin lassen sich mit einem gespeicherten Datensatz mehrere unterschiedliche Auswertungen vornehmen, was im Off-line-Fall nicht möglich ist - dort muß für jede Auswertung eine neue Messung durchgeführt werden.

3.6 Benutzerunterstützung

Bei Messungen in einem verteilten System sind alle Meßkomponenten zwangsläufig örtlich verteilt. Selbst in den Fällen, wo in Laborversuchen [70] alle Objektsysteme aufgrund des Synchronisationsverfahrens örtlich beieinander stehen, ist eine zentrale Steuerung aller Meßkomponenten sinnvoll.

Bezüglich des Grades der zentralen Kontrolle (siehe Bild 3.8) unterscheidet man in Ansätze, bei denen die verteilten Meßteile ohne eigene Kontrollfunktionen sind und demzufolge ferngesteuert werden (MASTER-SLAVE-Beziehung) bis hin zu den Ansätzen, in denen die verteilten Meßteile die Durchführung der Messung und deren Auswertung mitentscheiden.

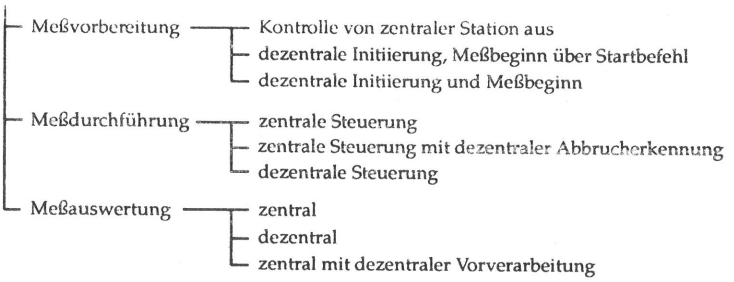


Bild 3.8: Steuerungsmöglichkeiten einer Messung

3.6.1 Meßvorbereitung

Bevor die eigentliche Messung durchgeführt wird, sind folgende Schritte durchzuführen:

- Verfügbarkeit der verteilten Meßkomponenten überprüfen,
- verteilte Meßkomponenten initiieren (Uhren und Speicher löschen),
- Meßpunkte konfigurieren (Meßpunkt auf Meßaufgabe anpassen),
- Meßpunkte aktivieren (Meßpunkt aktivieren),
- Objektprogramme laden und starten (sofern nicht bereits geschehen) und
- zentrales Auswertesystem konfigurieren (vorbereiten zur Messung und Auswertung).

Die Reihenfolge ist implementierungsabhängig und endet in einem BEREIT-Zustand, der erst mit der Auslösung der Messung durch den Startbefehl verlassen wird. Verteilte Meßkomponenten zu starten beinhaltet, daß nicht nur ein Startbefehl ausgesandt wird, sondern auch dessen Empfang auf allen an der Messung beteiligten Meßkomponenten geprüft werden muß. Nur so ist sichergestellt, daß die Messung korrekt begonnen wird. Kann eine Meßkomponente nicht ordnungsgemäß mit der Messung beginnen, so muß dies sofort erkannt werden, damit entweder diese Messung (mit veränderten Rahmenbedingungen) durchgeführt oder abgebrochen und neu gestartet wird.

Für gewöhnlich wird man die Messung von zentraler Stelle aus starten. Dies ist jedoch gleichwertig mit den Anwendungen, bei denen durch ein vordefinierbares Ereignis alle Meßkomponenten aus einem BEREIT-Zustand in den eigentlichen Meßzustand übergehen. Solche vordefinierbare Ereignisse (Neustart eines Objektsystems, spezielle Netzrahmen oder Netzauslastung oberhalb einer Schwelle) müssen im BEREIT-Zustand von allen Meßkomponenten gleichzeitig erkannt werden können. Das dezentrale Erkennen einer Startbedingung hat den Vorteil, daß der Aufwand zum zeitgleichen Aussenden des Startbefehls an alle entfällt. In den hier betrachteten lokalen Netzen besteht die Möglichkeit, einen Startbefehl in einem Rahmen mit Gruppenadresse zu versenden. Wird der korrekte Empfang dieses Rahmens durch je einen Quittungsrahmen eines Gruppenmitglieds bestätigt, so überfluten bei größeren Mitgliederzahlen die Quittungen den Sender, so daß spezielle gesicherte Protokolle notwendig werden.

3.6.2 Meßdurchführung

Für die Meßdauer arbeiten die Meßkomponenten selbständig, d.h. sie extrahieren die Informationen aus den Objektsystemen. Bei dezentralen Speichern ist nur

- eine Überwachung des bereits belegten Speichers und
- des internen Zustandes der verteilten Meßkomponenten

durch die zentrale Auswertestation sinnvoll. Die Abbruchbedingung kann von zentraler Seite zeit- oder ereignisgesteuert erfolgen. Dezentrale Bedingungen für einen Abbruch sind erkannte Fehler oder Speicherüberlauf in einer der beteiligten Komponenten. Deshalb prüft die zentrale Auswertestation periodisch die Meßteile ab, sofern diese nicht von sich aus eine Endemeldung versenden.

Speichert man die Meßinformationen an zentraler Stelle, so können in den dezentralen Meßteilen nur Fehler wie verlorene Ereignisse, Pufferüberläufe oder Uhrenüberläufe auftreten, die zusammen mit den Meßinformationen signalisiert werden. Daraufhin kann über Abbruch oder Fortgang der Messung entschieden werden.

3.6.3 Meßauswertung

Bild 3.8 zeigt, daß neben der zentralen Auswertung auch Auswertungen in den Meßteilen sinnvoll sein können. Damit läßt sich nicht nur die Belastung für die zentrale Auswertestation senken, sondern auch die zu transferierende Datenmenge reduzieren. Einfache Berechnungen wie Filterung, Häufigkeiten von Ereignissen, Summen von Laufzeiten oder Dekodierungen sind sogar in Realzeit realisierbar [143], führen aber zu einer Informationsreduzierung, wenn die zur Berechnung verwendeten Daten anschließend gelöscht werden.

Kann die statistische Auswertung nur an zentraler Stelle durchgeführt werden, so stellt die möglicherweise riesige Informationsmenge ein ernstzunehmendes Problem dar. [137] beschreibt, daß die reservierten 36 MBytes im Hauptspeicher des Objektsystems innerhalb von Minuten vollgeschrieben wurden. [143] beschreibt das Verbindungsnetzwerk zwischen den Meßteilen mit einer Bandbreite von 10 MBytes/s Meßdaten als möglichen Engpaß, was selbst bei kurzen Meßdauern im Sekundenbereich auf riesige Datenmengen schließen läßt.

Bei der Auswertung solch umfangreicher Datenmengen ist schnell die physikalische Kapazität des Benutzerspeichers auf dem Auswertesystem überschritten, so daß beim Suchen in den Datensätzen sehr intensives Auslagern von Speicherbereichen auf Hintergrundspeicher (engl. Swapping) nötig wird oder, falls dies der Auswerterechner nicht unterstützt, direkt auf die Ereignisse in den Datenfiles (und damit über langsame Systemroutinen) zugegriffen werden muß. Selbst wenn diese Datenmengen noch beherrschbar sind, so kann dies bei Durchführung einer Meßreihe zu Engpässen auf den Hintergrundspeichern des Auswerterechners führen.

4 Architektur für ein verteiltes Meßsystem

Zielsetzung beim Entwurf dieses neuen verteilten Meßsystems ist es, ein Konzept zu finden, das es erlaubt, mit minimalen Mitteln verteilte Messungen in realen Umgebungen durchzuführen, ohne den vorhandenen Netzverkehr zu beeinträchtigen.

Die reale Umgebung besteht hierbei aus integrierten Steuerungen bis hin zu leistungsfähigen Zellrechnern, die auf unterschiedlichsten Hardware-Plattformen aufbauen, auf denen unter verschiedenen Betriebssystemen die Kommunikationsprotokolle direkt oder über spezielle Netzzugangsbaugruppen implementiert sind.

Ein möglicher Einsatzort des verteilten Meßsystems ist eine sich in Betrieb befindliche Fabrikationsumgebung, die entsprechend der CIM-Philisophie alle Ebenen der Fertigung umfaßt. Daher scheidet die zusätzliche Verlegung eines Monitornetzes, aber auch eines separaten Synchronisationsnetzes aus. Die einzig möglichen Verbindungsstrukturen der einzelnen Meßkomponenten sind die aus Bild 3.5b und 3.5c. Diese beiden Strukturen entsprechen einem Software-Monitor (gezeigt in Bild 3.5b) und einem Hardware oder Hybrid-Monitor (Bild 3.5c). Da die Systembeeinträchtigung durch einen Software-Monitor auf das reale System zu groß ist, bleibt nur die Auswahl zwischen Hybrid- und Hardware-Monitor. Der Hardware-Monitor scheidet aus, da Messungen bis in die Anwendungsebene durchzuführen sind.

4.1 Hybridmonitor

Mit der Entscheidung für einen Hybrid-Monitor ergeben sich weitere Fragestellungen, die von der Testumgebung und einer potentiellen Einsatzumgebung abhängen. Hier wird ein Lokales Netz, basierend auf Bus-Topologie mit dem Kanalzugriffsverfahren CSMA/CD angenommen, obwohl aufgrund der MAP-Initiative das Token Passing-Zugriffsverfahren mit Bus-Topologie im Fertigungsbereich zu erwarten ist. Daß die hier getroffene Annahme des CSMA/CD-Protokolls auf Bus-Topologie zulässig ist, zeigen die von [61] in Fertigungsbetrieben durchgeführten Messungen: alle Netze basierten auf diesem Zugriffsverfahren. Das entwickelte Systemkonzept ist unabhängig vom jeweils verwendeten Lokalen Netz und wird in Kap. 4.3 noch ausführlicher erläutert.

Neben der Netztopologie und dem dort eingesetzten Zugriffsprotokoll hat auch die Speicher- und Zeitsynchronisationsproblematik Einfluß auf die Art der möglichen Messungen. Wie aus Bild 3.4 ersichtlich ist, verlangen die Verfahren zur Zeitsynchronisation a) bis c), daß eine Zeitinformation von zentraler Stelle an die örtlich verteilten Komponenten übertragen wird. Dies erfordert ein separates Netz zur Synchronisation oder die Nutzung des vorhandenen Nutznetzes, was obiger Forderung nach einer Minimallösung widerspricht. Auch das Verfahren nach Bild 3.4d belastet durch Software-Synchronisationsprotokolle das vorhandene Netz und scheidet hier aus. Als einzige Alternative verbleibt ein Synchronisationsverfahren basierend auf Analyse der aufgezeichneten Meßdaten.

Damit ergibt sich aber gleichzeitig die Notwendigkeit, daß alle erfaßten Meßdaten dezentral zwischengespeichert werden müssen. Zur abschließenden Offline-Auswertung aller Meßdaten ist ein Transfer zu einer zentralen Auswertestation unumgänglich, obwohl der dabei anfallende zusätzli-

che Netzverkehr das Nutznetz belastet. Diese Belastung am Meßende kann durch zufällig verzögertes Übertragen der zwischengespeicherten Meßdaten reduziert werden, so daß nicht alle an der Messung beteiligten Meßkomponenten ihre Meßdaten zur selben Zeit an die Auswertestation übertragen. Andererseits können durch sofortiges Auslesen der dezentralen Speicher die Meßdaten für eine laufende Auswertung an die Auswertestation transferiert werden, wenn die daraus resultierende Netzbelastung akzeptiert wird.

Der Anschluß der einzelnen Meßkomponenten an das reale Netz muß in der oben beschriebenen Umgebung durch einen eigenen Netzzugang möglich sein. Die Verwendung des Objektsystemzugangs an das Netz würde massive Eingriffe in die Hardware- und Software-Struktur des jeweiligen Systems bedeuten, während ein eigener Netzzugang zu einem Lokalen Netz heute mit intelligenten LAN-Koprozessoren [3, 90, 149, 182] nur wenig Aufwand verursacht.

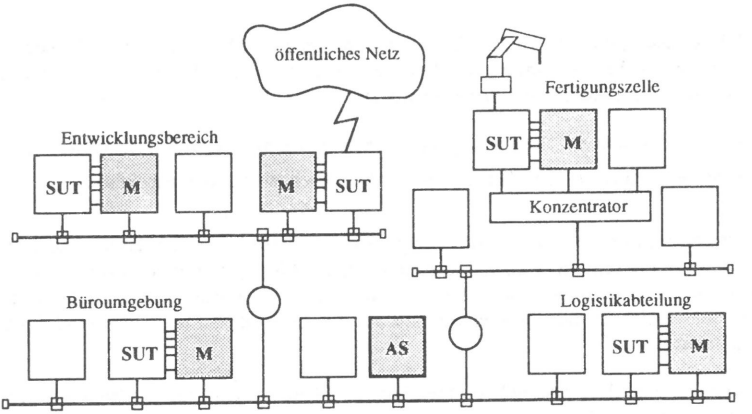


Bild 4.1: Netz mit verteiltem Hybrid-Monitor
(M Monitorkomponente, AS Auswertestation, SUT Objektsystem)

Bei erhöhtem Aufwand sind neben der im Bild 4.1 gezeigten Verbindungsstruktur aller Monitorkomponenten über das vorhandene Netz die in Bild 3.5d gezeigte Verbindung über ein zusätzliches separates Monitornetz oder Kombinationen davon denkbar. Der zusätzliche Aufwand für Netzanschlußeinheiten (engl. Attachment Units oder Taps) reduziert sich, wenn spezielle Konzentratoren (engl. Fan-Out Units), wie in Bild 4.1 gezeigt, eingesetzt werden.

Im Bild 4.2 gibt es zwei Klassen von Monitorkomponenten:

- Monitorkomponenten, die am vorhandenen Netz angeschlossen sind und über Brücken (engl. Bridges) erreichbar sind oder
- Komponenten, die am zusätzlichen Monitornetz direkt angeschlossen sind.

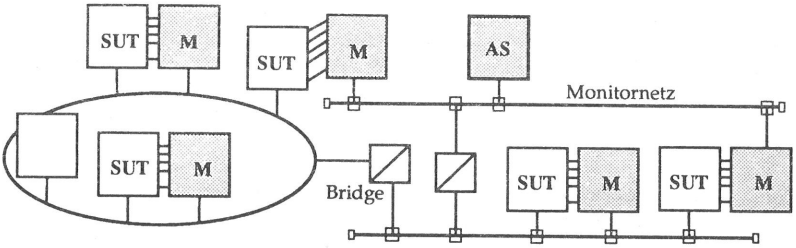


Bild 4.2: Hybrid-Monitor mit unterstützendem Monitornetz (SUT Objektsystem, M Monitorkomponente, AS zentrale Auswertestation)

Als Minimalausstattung werden die Objektsystem-abhängigen Meßfühler sowie die externen Monitorkomponenten gebraucht, um dezentral messen zu können. Diese Konfiguration kann durch weitere optionale Funktionseinheiten erweitert werden:

- Konzentratoren zur Verbindung mehrerer Stationen mit einem Netzanschluß und
- Massenspeicher zur lokalen Ablage von Meßdaten auf Diskettenspeicher.

Üblicherweise transferiert die Monitorkomponente alle Meßdaten über das vorhandene Netz zur Auswertung an die zentrale Station. Sollte dies aber im Störungsfall nicht möglich sein, so können die Meßdaten auch lokal auf Disketten gespeichert werden. Eine Auswertung der Meßdaten vor Ort ist nicht notwendig, so daß die Steuerung nur für die Kontrolle der einzelnen Funktionseinheiten dimensioniert wird.

Vom Konzept wird der Anschluß von Meßfühlern an unterschiedlichen Meßstellen unterstützt, um alle Aktionen des Objektsystems zu verfolgen:

- unmittelbarer Anschluß an das Netz,
- Meßpunkte an der Hardware und
- Meßpunkte in der gesamten Software.

Der Meßpunkt direkt am Netzzugang erlaubt die Beobachtung des gesamten von diesem System erzeugten und empfangenen Netzverkehr. Hierbei überlagern sich alle Verkehrsströme der auf diesem Objektsystem ablaufenden Applikationen sowie der von den einzelnen Kommunikationsschichten selbst erzeugte Netzverkehr. Der hier meßbare Rahmenverkehr gibt Aufschluß über die von diesem System mit seinen darauf ablaufenden Anwendungen erzielbare Verkehrsleistung bei gegebener Netzbelastung und in Abhängigkeit der beteiligten Kommunikationspartner.

Messungen, bei denen die Rate der Rahmen, deren Verteilung sowie die Kommunikationsströme gemessen werden, sind an dieser Meßstelle auch mit einem der kommerziell verfügbaren LAN-Tester [78, 79, 151, 184] durchführbar. Bei allen existierenden LAN-Testern kann jedoch nur ein Meßpunkt betrachtet werden, auch sind die Auswertungsmöglichkeiten beschränkt. Neben der Paketdekodierung zur Fehlerfindung dienen diese Meßgeräte zur einfachen Netzüberwachung und berechnen üblicherweise Mittelwerte und Klasseneinteilungen nach Rahmentypen, der Länge oder

dem zeitlichen Rahmenabstand. Ein weiterer Nachteil dieser LAN-Tester ist, daß sie als eigenständige Meßgeräte konzipiert sind: sie sind nicht fernsteuerbar und lassen sich deshalb nicht in dieses Meßkonzept einbauen.

Am Meßpunkt zwischen Netz und Objektsystem ist der gesamte Netzverkehr sichtbar. Es sollen jedoch vom dort angebrachten Meßfühler nur die Rahmen identifiziert werden, die vom Objektsystem erfolgreich über das Netz ausgesendet oder erfolgreich vom Netz empfangen werden. Mit dieser Forderung werden Eigenschaften des Übertragungskanals, des Zugriffsprotokolls und des Objektsystems wie

- Übertragungsfehler,
- zu kurze oder zu lange Rahmen,
- Rahmenkollisionen (bei CSMA/CD),
- Empfangsverluste durch Pufferengpässe oder Speicherzugriffskonflikte oder
- Erreichbarkeit des Objektsystems über verschiedene Netzzugänge oder Adressen

unsichtbar gemacht. Dies ist eine schärfere Forderung als die der LAN-Tester: dort genügt es, wenn der Rahmenkopf erkannt wurde, während hier nur der real transferierte Rahmen erfaßt wird.

Entsprechend Bild 4.3 lassen sich zusätzlich Meßpunkte in der Hardware des Objektsystems definieren. Dabei wird hier nicht unterschieden, ob die beobachtete Hardware Teil einer Netzzugangsbaugruppe ist oder unter Kontrolle des Betriebssystems läuft. Aus Aufwandsgründen bleiben die Meßpunkte auf wenige Abgriffstellen beschränkt:

- LAN-Koprozessor zur Abwicklung der Medienzugriffsprotokolle,
- Systembus der Netzzugangsbaugruppe und
- Systembus des Hostrechners.

Speziellere Meßpunkte (etwa am Festplatten-Kontroller, an Schnittstellenbausteinen zur Ein-/Ausgabe von Konsolen oder Speicherbaugruppen) sind denkbar, lassen sich aber einsparen, wenn dieselbe Information auch an einem der Systembusse gewonnen werden kann.

Ein Problem beim Entwurf des Hybrid-Monitors besteht darin, die Grenze zwischen den hardwaremäßig und den softwaremäßig realisierten Meßpunkten festzulegen. Der erforderliche Aufwand für Hardware-Meßfühler ist beachtlich, so daß deren Einsatzbereich auf die oben genannten drei Stellen begrenzt wird. Gerade die Systembusse sind die Stellen, über die in konventionellen Rechnern mit von Neuman-Architektur alle Daten transportiert werden. Damit lassen sich viele Meßpunkte, die hardwaremäßig erfaßbar wären, hier abgreifen: durch Kenntnis der Systemabläufe oder durch gezielte Modifikationen in der ablaufenden Software markiert man die Stellen, an denen eine Hardware-Aktion stattfinden würde. Ein Zugriff auf spezielle, sonst nicht belegte, Adressen oder der Transfer kennzeichnender Daten auf unbenutzte Ressourcen kann der Auslöser (engl. Trigger) für die Meßpunkterkennung sein. Dies vereinfacht die Meßpunkterkennung, da nur noch am Systembus ein Meßfühler installiert werden muß.

Letztendlich bleiben diejenigen Meßpunkte, die innerhalb der rein softwaremäßig implementierten Systemfunktionen liegen und am einfachsten durch Software-Modifikationen erkennbar sind. Diese Modifikationen stellen Software-Meßfühler dar und extrahieren die gewünschte Information

aus den laufenden Programmen. Anschließend muß diese Information an den Hybrid-Monitor übergeben werden, was aus Gründen der Effizienz und Kosten wieder über einen am Systembus aufgebrachten Hardware-Meßfühler geschieht.

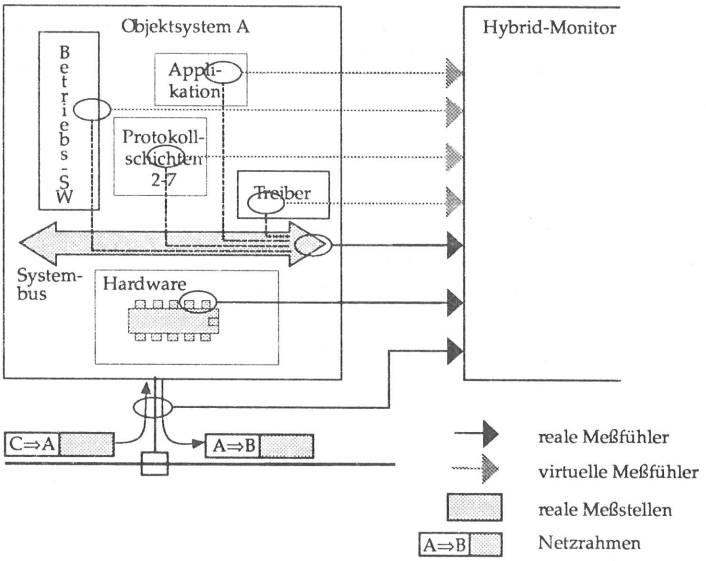


Bild 4.3: Realisierte Meßstellen

Dabei spielt es keine Rolle, woher die Informationen an den Meßpunkten stammen. Meßpunkte im Betriebssystem des Objektrechners initiieren ebenso spezielle Aktionen wie alle Meßpunkte in den Anwendungen, den Kommunikationsprotokollen oder den notwendigen Treibern, d.h. die virtuellen Meßfühler direkt an den zu messenden Softwareprogrammen lassen sich auf einen real vorhandenen Meßfühler am Systembus abbilden.

4.1.1 Hardware-Meßfühler

Vom Konzept des verteilten Hybrid-Monitors werden drei Typen von Meßfühlern unterstützt:

- am Netzzugang,
- am LAN-Koprozessor und
- am Bussystem.

4.1.1.1 Netzzugang

Durch Beobachtung der seriellen Datenströme, die das Objektsystem in das Netzkabel einspeist und von dort liest, lassen sich definierte Stellen in allen Netzrahmen untersuchen. Durch die Vorgabe des Musterbeginns innerhalb der Rahmen können beliebige Stellen wie MAC-Adreßfelder oder LSAPs gefunden werden. Die von diesen Meßfühlern erzeugte Information ist dann entweder das dort gefundene Muster selbst oder ein Indikator dafür, ob dieses Muster mit einem Referenzmuster übereinstimmt.

4.1.1.2 LAN-Koprozessor

Alle erhältlichen LAN-Koprozessoren sind als integrierte Bausteine ausgelegt, die selbständig die Abarbeitung des Medienzugangsprotokolls erledigen. Dabei gibt es drei Arten, wie die Ankopplung eines LAN-Koprozessors an den Prozessor (engl. Central Processing Unit, CPU) erfolgen kann.

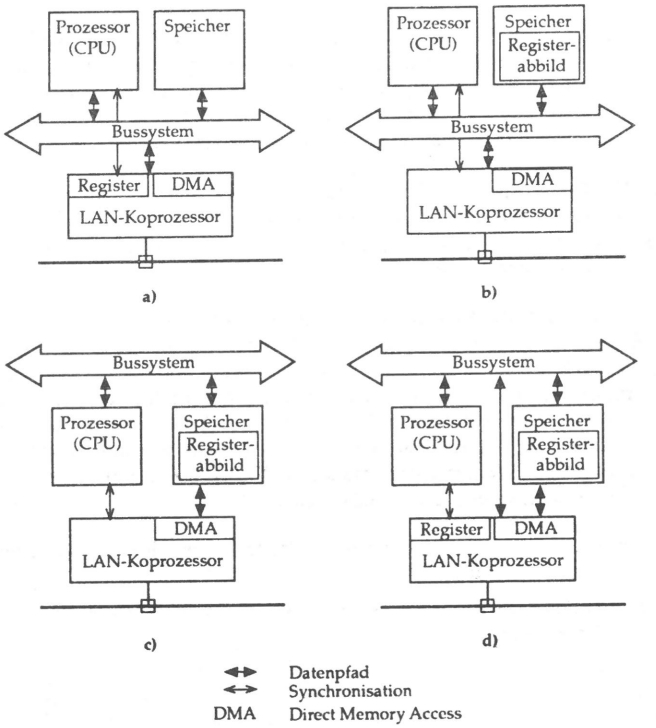


Bild 4.4: Ankopplung der LAN-Koprozessoren

In Bild 4.4 sind die möglichen Anbindungen skizziert:

- Zugriff der CPU auf die Register des LAN-Koprozessors über den gemeinsamen Bus (Bild 4.4a) [3, 149, 182],
- CPU hat keinen direkten Zugriff auf die Koprozessorregister, deshalb legt LAN-Koprozessor eine Kopie seiner internen Register im gemeinsamen Speicher ab (Bild 4.4b) [92],
- Zweitor-Speicher verbindet CPU und Koprozessor, in den der Koprozessor seine internen Register ablegen muß, damit die CPU diese auswerten kann (Bild 4.4c) [92] oder
- direkter Zugriff der CPU auf die wichtigsten Register direkt, alle weiteren Register im Speicherabbild (Bild 4.4d) [147].

Die zu transferierenden Daten werden bei allen LAN-Koprozessoren durch eine integrierte DMA-Einheit direkt im Speicher abgelegt. Die Synchronisation erfolgt im allgemeinen durch eine Initialisierungsmeldung der CPU zum Anstoß eines Rahmentransfers sowie durch eine Unterbrechungsmeldung des LAN-Koprozessors an die CPU bei Rahmenempfangsende.

Aufgrund der unterschiedlichen Ankopplungsarten gibt es keine einheitliche Stelle, an der passiv erkannt werden kann, ob Rahmen erfolgreich gesendet oder empfangen wurden. In Systemen gemäß Bild 4.4a kann durch Beobachtung der Registerzugriffe, die am Bus sichtbar sind, das Meßziel erreicht werden. Aufwendiger wird dies in den speichergekoppelten Systemen, bei denen der LAN-Koprozessor selbst die Registerinhalte im Speicher ablegt. Sind die Speicherzellen bekannt, an der das Registerabbild sich im Speicher befindet, so läßt sich wie im vorhergehenden Fall der Zugriff der CPU auf diese Speicherzellen am Bus beobachten. Schreibt dagegen der Koprozessor das Registerabbild auf wechselnde Speicherzellen (in Listen oder zu den jeweiligen Datenpaketen) (Bild 4.4c) oder sind die Register auf Koprozessor und Speicher verteilt (Bild 4.4d), so scheidet eine busseitige passive Beobachtung aus.

Der andere, hier gewählte, Ansatz legt die Beobachtungsstelle auf den Koprozessor. Bei Koprozessoren mit Registerzugriff entspricht dies einem Meßfühler am Bus. Falls der Koprozessor seine internen Register im Speicher ablegt, müssen nun diese Schreibbefehle erkannt und ausgewertet werden. Das Problem dabei ist, daß bei einer von Neuman-Architektur Schreibzugriffe von Paket- und Registerdaten nicht unterscheidbar sind. Dies verlangt, daß die Arbeitsweise des vorliegenden Koprozessors bekannt und deterministisch ist, so daß im Meßfühler das sequentielle Verhalten nachgebildet werden kann. Weiterhin ist es mit diesem Meßansatz nicht möglich, das von [22] definierte Meßverfahren anzuwenden. Dort wählte man den

- Absendezeitpunkt als Zeitpunkt, an dem das letzte Bit eines Rahmens die Meßschnittstelle abgehend passiert und den
- Ankunftszeitpunkt als Zeitpunkt, an dem das erste Bit des Adreßfeldes eines neuen Rahmens empfangen wird.

Zu diesen Zeitpunkten kann aber nocht nicht der korrekte Empfang oder Versand eines Rahmens festgestellt werden, weshalb für die hier gewählte Meßumgebung eines Lokalen Netzes als Meßpunkt die Verbindung des Koprozessors zum restlichen Objektsystem betrachtet werden muß.

4.1.1.3 Bussystem

Meßfühler am Busystem stellen die dritte Art der Hardware-Meßfühler dar. Falls eine eigene CPU nur die Steuerung des LAN-Koprozessors übernimmt, gibt es neben diesem lokalen Bus noch den eigentlichen Systembus, ansonsten ist der Koprozessor direkt am Systembus angeschlossen.

Hier können zentral alle abbildbaren Aktionen, die im Objektsystem ablaufen, erfaßt werden. Gebräuchliche Systembusse wie VMEbus, Multibus oder IBM AT-Bus verwenden heute einen parallelen Adreß- und 16bit oder 32bit Datenpfad, über den alle Baugruppen angeschlossen sind. Dabei gibt es unterschiedliche Zugriffsmodi:

- Speicherzugriffe,
- Zugriffe auf Ein-/Ausgabekanäle und
- DMA-Zugriffe.

DMA-Zugriffe sind eine spezielle Anwendung der beiden anderen Zugriffsmodi, bei denen mittels spezieller Bausteine größere Datenmengen als Block, als Bursts oder im Cycle Stealing-Verfahren übertragen werden. Die dabei verwendeten Speicherzugriffe sind auf Geschwindigkeit optimiert, während die Zugriffe auf Ein-/Ausgabekanäle meist langsamer erfolgen können. Zugriffsmodi zur Unterbrechungsbehandlung sind am Bus erlaubt, werden jedoch nicht verwendet.

Falls nicht die Erkennung der einzelnen Buszustände, sondern die über den Bus transferierten Daten von Interesse sind, so genügt es, wenn der Meßfühler nur als eine Speicherstelle im Objektsystem erscheint. Alle signifikanten Daten liegen somit an dieser Speicherstelle an, die als *Monitor-Register* bezeichnet wird (vergl. Bild 4.5).

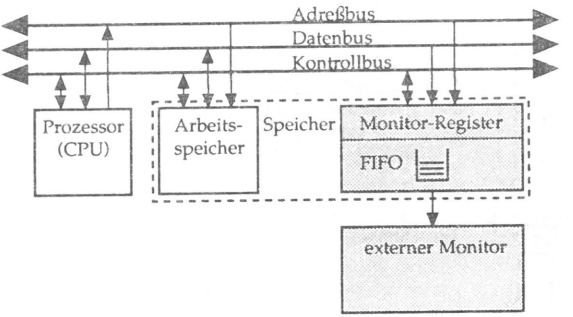


Bild 4.5: Monitor-Register als Hardware-Meßfühler am Systembus

In kleineren Systemen existiert ein solches Monitor-Register nicht, so daß dieses als Systemerweiterung erst in das Objektsystem eingebaut werden muß, bevor Meßdaten darüber an den externen Monitor weitergegeben werden können.

Prinzipiell gibt es mehrere Möglichkeiten, wie dieses Register zum Anschluß des Monitors bereitgestellt werden kann:

- Reservierung einer vorhandenen Speicherstelle oder eines Speicherblockes,
- Reservierung eines nicht verwendeten Ein-/Ausgabekanals,
- Verwendung von Speicherlücken, für die eine Adreßdekodierung vorhanden ist oder
- Einbau eines spezifischen Monitoranschlusses.

Die ersten beiden Methoden setzen voraus, daß in jedem zu beobachtenden System eine solche Reservierung möglich ist. Dies kann bedeuten, daß entweder unbenutzte und bekannte Speicherstellen im System vorhanden sind oder dem System eine vorhandene Speicherstelle entzogen werden muß. Analoges gilt, falls anstelle von Speicherstellen Ein-/Ausgabekanäle verwendet werden.

Andererseits gibt es in vielen Systemen Bereiche, deren Adressen zwar korrekt ausdekodiert werden, jedoch von der vorhandenen Hardware nicht belegt werden. Bei Zugriffen auf solche Speicherbereiche laufen die Buszyklen korrekt ab: die Daten werden jedoch nirgends abgelegt und ein Lesezugriff liefert unbestimmte Datenwerte zurück.

Die ersten drei Methoden sind sehr systemspezifisch und lösen noch nicht das Problem der Datenübergabe vom Objektsystem an den Monitor. In jedem dieser Fälle muß zusätzlich eine physikalische Verbindung zwischen Objektsystem und Monitor bereitgestellt werden.

Die letzte Methode beinhaltet bereits diese Verbindung. Das Monitor-Register selbst kann dann als neue zusätzliche Speicherstelle oder Ein-/Ausgabekanal implementiert sein. Wenn dieses Monitor-Register vom System nicht als verwaltete Ressource erkannt wird, steht somit eine Schnittstelle zwischen Objektsystem und Monitor zur freien Benutzung offen.

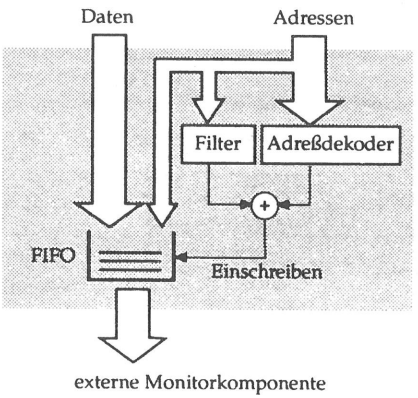


Bild 4.6 Informationsfluß im Bussystem-Meßfühler

Mit einem solchen Meßfühler lassen sich alle Signale am Systembus mitverfolgen. Sinnvollerweise reduziert man die extrahierten Informationen auf Adressen und Daten. Der größte Adreßteil wird zur eindeutigen Dekodierung der Adresse des Monitor-Registers im Adreßraum des Objektsystems benützt. Der verbleibende Teil der Adressse dient zur Kennzeichnung der geschriebenen Datenwerte und wird zusammen mit den Datenbuswerten an den Monitor übertragen. Neben der Kenn-

zeichnung wird gleichzeitig eine Filterfunktion, basierend auf der Adreßkennung, ausgeführt. Dazu wird jeder Adresse innerhalb des Monitor-Registerbereichs eine Maske zugeordnet, die eine Übernahme der auf diese Adresse geschriebenen Daten erlaubt oder sperrt.

Zur Entkopplung des Objektsystems vom externen Monitor dient ein FIFO-Speicher. Der schreibende Partner (Objektsystem) kann dann mit maximaler Geschwindigkeit seine Zugriffe ausführen, ohne daß auf eine erfolgreiche Datenübernahme im Monitor gewartet werden muß.

Für einen vielseitigen Einsatz ist es sinnvoll, wenn der Adreßbereich, unter dem das Monitor-Register angesprochen wird, ladbar ist. Dasselbe gilt für die einfache Adreßfilterung, die für unterschiedliche Messungen dynamisch konfigurierbar ist.

4.1.2 Software-Meßstatements

Es kommt das von [197] beschriebene Verfahren mit Monitor-Register zum Einsatz. Hierzu simuliert der Hardware-Meßfühler am Bus eine Speicherstelle, die nur beschrieben werden darf. Von der im Objektsystem ablaufenden Software sieht dieses Monitor-Register wie eine Speicherzelle aus. Als Konsequenz folgt, daß die zu erkennenden Aktionen in Schreibzugriffe auf ein Monitor-Register abgebildet werden.

Im Normalfall liegen fertige Programme vor, so daß für die einzubauenden Meßstatements die Implementierungssprache vorgegeben ist. Weiterhin kann auf die automatische Erzeugung der Meßpunkte verzichtet werden, da für die zu messenden Softwareteile meist keine Spezifikationen in einer Form vorliegen, aus denen die Software erzeugbar wäre. Somit liegt es beim Benutzer, welche Granularität er wählt:

- Befehle (engl. Statement),
- Blöcke,
- Prozesse oder
- Programme.

Eine Laufzeitmessung von einzelnen Befehlen scheitert an der Tatsache, daß die Implementierungssprachen wie C oder PL/M selbst zur Generierung der Wertzuweisungen verwendet werden: der Einfluß des Meßstatements kann nicht mehr abgeschätzt werden und kann größer sein als der zu messende Befehl selbst. Sofern mehrere Befehle, ganze Funktionen, Prozesse oder Programme gemessen werden, eignet sich dieser Ansatz.

In einem einfachen Betriebssystem wie MS-DOS kann der Benutzer jederzeit alle Speicherstellen unter ihrer realen Adresse ansprechen. Eine Eigenschaft von Betriebssystemen mit virtuellem Speicherkonzept jedoch ist es, daß der Benutzer nur mit logischen Adressen, nicht jedoch mit realen Adressen, rechnet. Dies verlangt, daß die bekannte Adresse des Monitor-Registers bei diesen Betriebssystemen in eine gültige logische Adresse übersetzt werden kann. Das Echtzeit-Betriebssystem iRMX II [94] läuft zwar im virtuellen Modus des Prozessors, stellt aber Systemaufrufe für eine Übersetzung der realen in logische Adressen bereit. Unter UNIX [181] gibt es aus Sicherheitsgründen keine solchen Systemaufrufe, so daß diesbezügliche Erweiterungen am Betriebssystem selbst nötig werden.

Die Informationen, die mit einem Software-Meßstatement in das Monitor-Register geschrieben werden, können ein oder viele Worte in Reihe sein und werden vom Benutzer festgelegt. Werden mehrere Worte geschrieben, so erkennt das Monitor-Register nicht, welche Worte als zusammengehörig betrachtet werden sollen, da jeder Schreibbefehl am Systembus als ein Speicherzyklus abgewickelt und als ein Informationswert an den externen Monitor übergeben wird.

Speicherzyklen, die Werte auf ausmaskierte Adreßstellen schreiben, laufen ohne Beeinflussung weiter. Der geschriebene Wert geht allerdings verloren. Da die Filterung über die Adreßzuordnung erfolgt, kann auf eine softwaremäßige Abfrage, ob ein betreffender Meßpunkt Information generieren soll, verzichtet werden.

4.1.3 Externe Monitorkomponente

Nachdem mit den Hardware- und Software-Meßfühlern die vom Objektsystem abhängigen Teile beschrieben wurden, wird nun der interne Aufbau der Monitorkomponente genauer beschrieben.

Aufgabe dieser Komponente ist es, den von einem im Objektsystem installierten Meßfühler generierten Informationswert zu empfangen, zu filtern und, zusammen mit dem Auftrittszeitpunkt, im Speicher abzulegen. Dabei wird jeder Informationswert als ein Wert innerhalb des Informationsstroms einer Meßphase betrachtet. Alle für diesen Informationswert anfallenden Bearbeitungsschritte sind in der Zeit vom Empfangen bis zum Ablegen im Speicher durchzuführen. Eine gleichzeitige Auswertung der Informationsinhalte oder die Mitverfolgung von mehreren Informationswerten, die zusammen eine Ereignisstruktur bilden, ist nicht enthalten.

Da aufgrund der oben genannten Anforderungen der Meßdatenspeicher und die Zeitbasis in jeder Monitorkomponente vorhanden sein müssen, ergibt sich die in Bild 4.7 abgebildete Struktur.

Jede externe Monitorkomponente besteht aus den vier wesentlichen Funktionseinheiten

- Abfragestufe — FIFO-Speicher und periodische Abfrage aller Meßfühler,
- Zeitbasis — basierend auf Oszillator und Zähler,
- Meßspeicher — zur Ablage der Meßdaten samt Zeitstempel und
- Steuerung — zur Koordination der Monitorkomponente.

4.1.3.1 Abfragestufe

Die Abfragestufe bildet das Verbindungsstück zwischen den im Objektsystem installierten Meßfühlern und dem externen Monitor. Die in Bild 4.7 eingezeichneten FIFO-Speicher entsprechen den FIFO-Speichern, die bei den Meßfühlern zur Geschwindigkeitsanpassung beschrieben wurden. Alle Meßfühler sind über einen synchronen parallelen Bus mit der Monitorkomponente verbunden. Durch Abfrage (engl. Polling) der einzelnen Meßfühler, d.h. durch Auslesen von Infor-

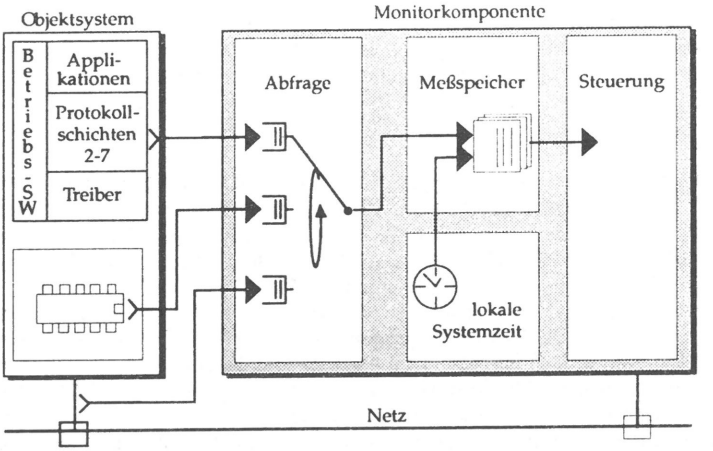


Bild 4.7: Interne Struktur einer Monitorkomponente

mation aus den zugehörigen FIFO-Speichern, werden die geschriebenen Informationen empfangen und zusammen mit der in diesem Zeitpunkt gültigen Zeitinformation aus der lokalen Zeitbasis im Meßspeicher abgelegt.

Die Anzahl der anschließbaren Meßfühler ist durch die gewählte periodische Abfragebetriebsart beschränkt: alle Meßfühler werden innerhalb jedes Zyklus genau einmal auf einen vorliegenden Informationswert abgefragt und legen damit bei implementierungsabhängiger Verweilzeit die maximal erreichbare Abfragerate fest. Alle zwischen Abfragezeitpunkten im FIFO eintreffenden Informationswerte warten bis zum nächsten Abfragetermin. Werden die Informationswerte schneller in die FIFO-Speicher eingeschrieben als ausgelesen, so bildet sich im FIFO eine Schlange von wartenden Informationswerten, die in aufeinanderfolgenden Zyklen ausgelesen und zeitlich markiert werden. Damit dieser Zustand nicht eintritt, erfolgt das Auslesen mit einer Rate, die vom Objektsystem nicht erreicht werden kann.

Alle Meßfühler am Bussystem oder am LAN-Koprozessor werden gleichberechtigt abgefragt, eine nichtperiodische Abfertigungsstrategie ist nicht nötig, da ein Abfragezyklus kürzer ist als der Abstand zwischen zwei mit maximaler Rate erzeugten Ereignisse.

4.1.3.2 Lokale Systemzeit

Trotzdem kann es vorkommen, daß Informationswerte von zwei Meßfühlern zur gleichen Zeit eintreffen. Durch die sequentielle Abfrage aller Meßfühler werden diese Informationswerte jedoch nacheinander ausgelesen und zeitlich markiert. Dazu wird der aktuelle Uhrenstand der lokalen Zeitbasis gelesen, der mit jedem Abfragezyklus inkrementiert wird, d.h. die Zeitinformation ist real ein Zykluszählerstand. Somit besagt ein Zeitwert nur, in welchem Abfrageintervall ein Informationswert von einem Meßfühler vorgelegen hat.

Die Basis der Uhr ist ein Oszillator, der frei bei bekannter Frequenz schwingt. Zur Anpassung auf das aktuelle Meßproblem kann dieser Oszillator auf verschiedene Frequenzen eingestellt werden,

wodurch sich eine veränderte Zykluszeit und damit eine abgeleitete Abfragerate ergibt. Höhere Abfrageraten erlauben häufigere Abfragen der FIFO-Speicher und kürzere Meßphasen, während niedrige Abfrageraten eine geringere Informationsrate unterstützen zugunsten längerer Meßphasen.

4.1.3.3 Meßspeicher

Neben der aus dem Objektsystem empfangenen Information und dem Zeitstempel wird noch die Kennung der Abfragestelle innerhalb des Abfragezyklus mit abgespeichert. Jeder Eintrag in den Meßspeicher erfolgt als ein 56 bit breites Speicherwort bestehend aus 16 bit Informationsfeld, 3 bit Meßfühleradresse und 32 bit Zeitstempel. Als Organisationsform dient das FIFO-Prinzip, wobei die nötige Einschreibebreite, die möglichst große FIFO-Länge sowie die angestrebte Abfragerate mit kommerziellen Bauelementen nicht zu realisieren sind. Ein weiteres Problem stellt sich beim Auslesen des FIFOs. Da das Einschreibewort deutlich breiter als die von der Steuerung unterstützte Busbreite ist, muß im FIFO beim Auslesen eine Serialisierung stattfinden. Als weitere Speicherform dient der zyklische Ringpuffer, bei dem, im Gegensatz zum FIFO-Prinzip, der vorhandene Speicher je nach Meßdatenaufkommen mehrfach beschrieben werden kann. Die dabei überschriebenen Speichereinträge gehen verloren, wenn sie durch die Steuerung nicht rechtzeitig ausgelesen und (z.B. unter Einsatz der Massenspeichererweiterung auf Disketten) gesichert werden.

4.1.3.4 Steuerung

Die Aufgaben der Steuerung in der dezentralen Monitorkomponente sind:

- Steuerung der funktionalen Einheiten (Abfragestufe, Zeitbasis und Meßspeicher),
- Steuerung aller optionalen Einheiten,
- Bereitstellung von Kommunikationsdiensten über LAN und lokalen Anschluß einer Bedienkonsole,
- Kommunikation mit der Auswertestation.

Es ist wünschenswert, daß neben dem eigentlichen Betrieb über das Netz zusätzlich ein Lokalbetrieb über eine anschließbare Konsole möglich ist. Diese Forderung erlaubt es auch, nur eine Monitorkomponente für Messungen zu verwenden. Andererseits können in den Fällen, wo verteilte Messungen aufgrund eines Netzproblems nicht ordnungsgemäß abgeschlossen werden, durch den lokalen Zugang die dezentral vorliegenden Daten gesichert werden. Zu diesem Zweck dient auch die optionale Speicherung der Meßdaten auf Hintergrundspeicher (Disketten), die sich an der Auswertestation wieder einlesen lassen.

Das auf der Steuerungsbaugruppe residente Betriebssystem erlaubt es, mehrere Applikationen gleichzeitig zu betreiben. Da je Objektsystem eine Monitorkomponente geplant ist (Bild 4.8a), ist immer nur eine Meßapplikation aktiv. Allerdings lassen sich mit einer Monitorkomponente mehrere Meßfühler anschließen (Bild 4.8b) und damit durch den Einsatz mehrerer Meßfühler auch mehr als ein Objektsystem beobachten (Bild 4.8c).

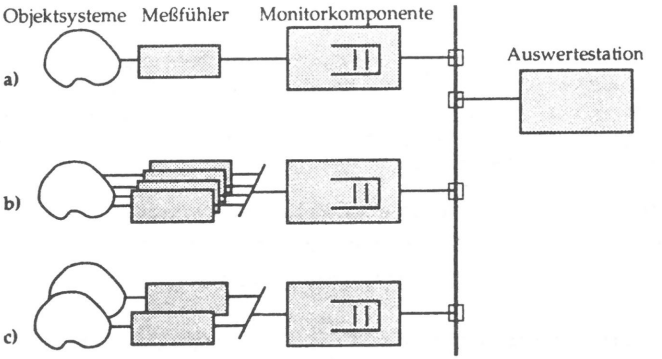


Bild 4.8: Verbindungsmöglichkeiten zwischen Monitor und Objektsystemen

Neben der Überwachung der einzelnen Funktionseinheiten ist die Bereitstellung von Kommunikationsdiensten die Hauptaufgabe der Steuerung. Alternativ zu einer lokal angeschlossenen Bedienerkonsole kann von der Auswertestation über das Netz selbst die Kontrolle durchgeführt werden. Über diese Kommunikationsschnittstelle wird die Meßkomponente über eine Befehlssprache gesteuert, die

- die Initialisierung der Meßfühler, des Speichers und der Zeitbasis vor der Messung,
- die Überwachung von Speicher und Zeitbasis während der Messung sowie
- das Auslesen der gemessenen Daten und deren Transfer auf Hintergrundspeicher, zum Benutzer an lokaler Konsole oder über das Netz

erlaubt. Daneben sind Regiebefehle zum Rücksetzen und Abbrechen einer Messung, zur Statusabfrage und zum Löschen des Speichers notwendig, um Fehlerzustände in örtlich entfernten Meßkomponenten von einer zentralen Stelle aus beheben zu können.

4.2 Zentrale Meßsteuerung

4.2.1 Master-Slave-Konfiguration

Im verteilten Meßsystem startet der Benutzer alle Meßaufträge von der zentralen Auswertestation aus. Es wird deshalb eine Konfiguration gewählt, bei der alle örtlich verteilten Meßkomponenten von der zentralen Stelle aus kontrolliert werden (Master-Slave-Betrieb). Dazu wird festgelegt, daß jede Meßkomponente die empfangenen Befehle auswertet und mit einer entsprechenden Rückmeldung quittiert. Erst danach folgt von der zentralen Station aus eine weitere Befehlsfolge. Selbständig erzeugte Meldungen von einer Meßkomponente an die zentrale Station sind nicht vorgesehen. Allein im Falle einer dezentral beendeten Messung darf eine Meldung von der Meßkomponente erzeugt werden, um die sonst notwendigen wiederholten Zustandsabfragen an alle Meßkomponenten einzusparen.

4.2.2 Kommunikationsprotokoll

Für den Transfer der zu übertragenden Befehle und Daten ist eine sichere Übertragung, unterstützt durch ein fehlertolerantes Protokoll, über das Lokale Netz nötig, das unter Verwendung bestehender Dienste auf Basis von ISO-OSI Kommunikationsprotokollen abgewickelt wird. Die unterste mögliche Schicht stellt die Sicherungsschicht (engl. Logical Link Control, LLC) entsprechend ISO 8802-2 dar. In diesem Standard sind drei Typen definiert:

- Type 1 Verbindungsloser Datagrammbetrieb,
- Type 2 Verbindungsorientierter Betrieb und
- Type 3 Quittierter verbindungsloser Datagrammbetrieb.

Aufbauend auf diesen drei Typen stellt [98] vier Klassen (engl. Classes) bereit (Tabelle 10).

Type	Class I	Class II	Class III	Class IV
1	•	•	•	•
2		•		•
3			•	•

Tabelle 10: Klassen in der Sicherungsschicht nach [98]

Für einen sicheren Transfer erlauben die Klassen LLC Class II bis IV über die Typen 2 und 3 entweder den verbindungsorientierten oder quittierten Betrieb zum sicheren Austausch von Rahmen

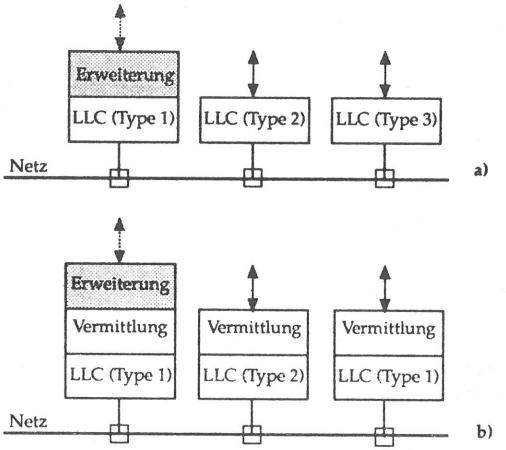


Bild 4.9: Protokollsäule zur Kommunikation zwischen den Meßteilen: a) innerhalb eines Lokalen Netzes, b) über unterschiedliche Netze

zwischen zwei an demselben physikalischen Netz angeschlossenen Systemen. Andererseits hat LLC Class I den Vorteil, daß es in jeder Klasse enthalten ist, jedoch durch eine zusätzliche, nicht OSI-konforme, Erweiterung noch fehlersicher gemacht werden muß (Bild 4.9).

Die Kommunikation zwischen den Meßkomponenten und der Auswertestation auf Basis von LLC verlangt, daß alle Meßteile an einem Lokalen Netz angeschlossen sind. Sind mehrere Lokale Netze über Brücken gekoppelt, können auch unterschiedliche Medienzugriffsverfahren in den einzelnen Netzteilen eingesetzt werden. Sobald eine Meßkomponente an einem anderen Netz angeschlossen wird, muß die Protokollsäule über eine Vermittlungsschicht verfügen (Bild 4.9b). Eine Schichtung gemäß Bild 4.9b hebt die Beschränkung der Kommunikationsmöglichkeiten auf ein physikalisches Netz auf, erfordert jedoch ein geändertes Zeitstempel-Resynchronisationsverfahren. Dieser Fall wird im folgenden nicht weiter betrachtet.

4.3 Synchronisation

4.3.1 Lokale Zeiterfassung

Die Zeiterfassung geschieht in den Monitorkomponenten, indem jede im FIFO-Speicher wartende Information beim Einschreiben in den Meßspeicher mit dem aktuellen Zeitwert markiert wird. Der Zeitwert entspricht der Anzahl von Abfragezyklen, die seit Beginn der Messung durchgeführt wurden. Jeweils zu Ende eines Abfragezyklus aller Meßfühler wird die Uhr (Zyklenzähler) um einen Zählerwert inkrementiert. Durch die taktgesteuerte Abfragebetriebsart kann aus der Taktperiode auf die Zeit bis zum Beginn des Abfragezyklus geschlossen werden. Beeinflussende Größen sind:

- Zeit zur Abfrage eines Meßfühlers,
- Zeit für Weiterschalten zum nächsten Meßfühler und
- Anzahl von Meßfühlern je Abfragezyklus,

da mit jeder Meßfühlerabfrage genau ein FIFO-Eintrag gelesen wird.

Wenn man vorgibt, daß jeder Abfrageschritt (unabhängig davon, ob ein Informationswert im FIFO-Speicher vorliegt) dieselbe Zeit beansprucht und bereits die Umschaltzeit beeinhaltet und die Anzahl abgefragter Meßfühler je Messung konstant ist, so berechnet sich die Erfassungszeit T_E zu:

$$
\begin{aligned}
T_E &= [T_{Eu}, T_{Eo}) \\
T_{Eu} &= (ZS * n + ID) * T_M \\
T_{Eo} &= ((ZS + 1) * n + ID) * T_M
\end{aligned}
\qquad \text{(Gl.2)}
$$

wobei:

ZS	Zeitstempelwert
T_M	Abfragezeit je Meßfühler
ID	lfd. Nummer des Meßfühlers
n	max. Anzahl abgefragter Meßfühler je Zyklus
T_{Eu}	untere Grenze der Erfassungszeit
T_{Eo}	obere Grenze der Erfassungszeit

Dieser so berechnete Wert gibt nur den von der Monitorkomponente sichtbaren Erfassungszeitpunkt an. Wann die Information in den FIFO-Speicher geschrieben wurde, ist nicht ersichtlich. Falls sich im FIFO-Speicher eine Warteschlange mit s Einträgen gebildet hat, war der reale Auftrittszeitpunkt mindestens s Abfragezyklen früher.

Durch die unabhängige Zeitmarkierung der Meßinformationen je Objektsystem [121] kann keine genaue Zuordnung der Ereignisspuren durchgeführt werden. Genaue Zeitzuordnungen, basierend auf den real verschickten Rahmen, sind nicht möglich, da beim Entwurf der Meßfühler am Netzzugang der erfolgreiche Transfer von Rahmen über den LAN-Koprozessor betrachtet wird und die Transfermeldung nicht deterministisch nach Empfang oder Versand eines Rahmens erfolgt.

4.3.2 Synchronisationsunterstützung

Der Aufwand für den Austausch von Zeitmeldungen oder ein separates Netz zur Verteilung der Zeitinformation kann entfallen, wenn über die gemessene Information selbst die Synchronisation erfolgt. Damit die Meßgenauigkeit besser wird als es mit dem Verfahren von [121] möglich ist, werden nicht die Rahmen an die betreffende Station sondern spezielle Rahmen (im folgenden als *SYNC-Rahmen* bezeichnet) auf dem Netz betrachtet. Zur Vereinfachung wird im folgenden angenommen, daß nur ein Absender diese SYNC-Rahmen während der Dauer einer Messung verschickt (was neben der einfacheren Realisierung auch weniger Aufwand bei der Resynchronisation erfordert).

Das gewählte Synchronisationsverfahren ist dann unabhängig vom Netz oder Zugriffsprotokoll wie CSMA/CD oder Token-Verfahren, wenn ein SYNC-Rahmen von allen am Netz angeschlossenen Monitorkomponenten erkannt werden kann. Dies ist in allen Lokalen Netzen der Fall, die an einem physikalischen Netz angeschlossen sind. Koppelt man einzelne Ringe oder Segmente über Bridges oder Brouter, so findet eine Filterung der Rahmen nach Adressen statt, so daß ein Rahmen nicht mehr von allen angeschlossenen Stationen empfangen werden kann. Bei derart gekoppelten Netzen kann durch die Wahl einer geeigneten Adresse (Gruppen- oder Broadcast-Adresse) erzwungen werden, daß der SYNC-Rahmen dennoch an allen Stationen empfangen werden kann.

Dieser zusätzliche SYNC-Rahmen dient dazu, daß in jeder Monitorkomponente bei dessen Empfang ein Eintrag mit spezieller Kennung (SYNC-Ereignis) in die lokale Ereignisspur stattfindet. Netzweit werden somit in allen dezentral gehaltenen Ereignisspuren, abhängig von der Lage innerhalb des Netzes, diese SYNC-Rahmen erkannt und mit ihrem Auftrittszeitpunkt abgespeichert.

Neben den SYNC-Rahmen, die mit einem geeigneten Sensor hardwaremäßig erkannt werden müssen, wird ein gleichzeitiger Transfer von Befehlen an die Steuerungen aller an einer Messung beteiligten Monitorkomponenten unterstützt. Die Komponenten sind dabei über das Netz unter einer Gruppenadresse (engl. Multicast Address) erreichbar, so daß durch Transfer eines einzigen Rahmens von der Auswertestation an alle beteiligten Monitorkomponenten am LAN Befehle übertragen werden können. Damit die Übertragung sichergestellt ist, wird vom verwendeten Kommunikationsprotokoll die Rückmeldung in Form einer Quittung verlangt.

Beginnen die Ereignisspuren zu unterschiedlichen Zeiten, so erschwert dies den Zuordnungsprozeß. Ein gleichzeitiger Beginn aller Monitorkomponenten wird gewährleistet, wenn alle Steuerungen synchron den Startbefehl erhalten. Wird der Startbefehl mit einer Multicast-Adresse an alle

Mitglieder der Messung versandt, so empfangen diese laufzeitverschoben diesen Befehl. Die Laufzeit ist durch die Netzkonfiguration vorgegeben, so daß dies bei der Zuordnung später Berücksichtigung findet. Als Unsicherheit verbleibt die dezentrale Bearbeitungszeit, die in den Monitorkomponenten unterschiedlich sein kann. Diese Bearbeitungszeit entfällt, wenn der Rahmen mit dem Startbefehl direkt als Auslöser einer Messung dienen kann. Die Erkennungslogik für die SYNC-Rahmen kann dazu dienen, daß die startbereiten Monitorkomponenten mit dem Empfang des ersten SYNC-Rahmens die Messung beginnen. Somit ist es möglich, daß ein beliebiges System am LAN mit dem ersten SYNC-Rahmen die verteilte Messung anstößt.

4.3.3 Resynchronisation

Die SYNC-Rahmen und der Multicast-Transfer sind Voraussetzung für eine im Anschluß an die Messung stattfindende erfolgreiche Resynchronisation der einzelnen Ereignisspuren. Da sich die Lage der Monitorkomponenten im Netz während der Messung nicht ändert, bleibt die Laufzeit zwischen einzelnen Monitorkomponenten konstant und wird beim Zusammenfügen der einzelnen Ereignisspuren zu der globalen Ereignisspur berücksichtigt.

Die Resynchronisation wird in drei Schritten durchgeführt. Zuerst müssen SYNC-Ereignisse einander zugeordnet werden, damit anschließend die Zeitachsen der frei laufenden Uhren durch Mittelwertbildung korrigiert werden können. Abschließend werden die mit interpolierten globalen Zeitstempeln markierten Informationen in einer globalen Ereignisspur (ohne SYNC-Ereignisse) nach ihrem Zeitstempel sortiert.

Das Zuordnen von sich entsprechenden SYNC-Ereignissen in der Ereignisspur kann auf unterschiedliche Weise erfolgen:

- direkte Zuordnung aller SYNC-Ereignisse (1.SYNC-Ereignis aus $Spur_1$ entspricht 1.SYNC-Ereignis aus $Spur_2$, usw.),
- zeitliche Abstände der SYNC-Ereignisse werden verwendet, um passende SYNCs in einzelnen Spuren zu finden oder
- SYNC-Ereignisse enthalten weitere Information wie die Rahmenlänge oder Rahmeninhalte, die als Zuordnungshilfe dienen.

Die einfachste Methode über direkte Zuordnung scheidet dann aus, wenn Rahmenverlust im Netz möglich ist, d.h. wenn einige Stationen den SYNC-Rahmen korrekt empfangen haben und andere Stationen nicht.

Aufwendiger ist die Zuordnung über den zeitlichen Abstand zwischen den SYNC-Rahmen. Dazu wird eine Folge gebildet, deren Werte aus den zeitlichen Abständen zwischen zwei SYNC-Ereignissen errechnet werden:

$$F^i = (T_1^i, T_2^i, T_3^i \ldots) \qquad \text{(Gl.3)}$$

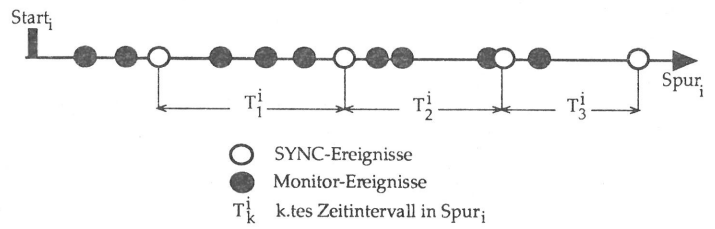


Bild 4.10: Zuordnung der SYNC-Ereignisse über Zeitabstände

Da aufgrund des Abfragebetriebs der einzelnen Meßfühler in der Monitorkomponente auch die Daten des zur SYNC-Rahmenerkennung eingesetzten Meßfühlers erst verzögert abgefragt werden und die dezentralen Uhren frei laufen, sind die aus unterschiedlichen Ereignisspuren errechneten Zeitabstände nicht konstant. Deshalb ist eine ON-LINE Zuordnung über die Zeitabstände nicht erfolgreich. Zusätzlich sorgen verlorene SYNC-Ereignisse für falsche Zeitabstände, die eine Zuordnung unmöglich machen.

Eine Zuordnung ist möglich, wenn die Ähnlichkeit der SYNC-Zeitabstände in den Ereignisspuren ausgenutzt wird. In Abhängigkeit des zeitlichen Abstands der SYNC-Rahmen eignen sich folgende Verfahren zur Ähnlichkeitsberechnung:

- Neuronale Netze,
- Korrelation oder
- diskrete Faltung.

Neuronale Netze eignen sich besonders zur Mustererkennung, bei der die Abbildung einer Eingabe auf die Ausgabe nicht beschrieben werden kann. Ein Neuronales Netz lernt durch Vorgabe von Referenzmustern diese Abbildung, die dann Bestandteil seiner Architektur (in Form von Gewichtungen und Schaltschwellen) wird [127]. Sind die zeitlichen Abstände einer Referenzfolge im voraus bekannt, so kann ein Neuronales Netz dieses Muster oder Teile davon lernen. Später lassen sich die aus den Ereignisspuren extrahierten SYNC-Ereignisabstandsfolgen (F^i) einzeln als Eingabe an das neuronale Netz geben. Durch Wegstreichen von Abstandswerten kann die Folge zum Referenzmuster verschoben werden, bis ein Maximum gefunden ist. Dieser Prozess wird für alle Ereignisspuren durchgeführt, um alle relativen Verschiebungen zu finden.

Dies ist dann durchführbar, wenn die erzeugte SYNC-Abstandsfolge immer den gleichen Verlauf hat. Wird der Strom der SYNC-Rahmen jedoch durch eine am Netz angeschlossene Station erzeugt, so kann dieses Muster abweichen: das Neuronale Netz ist jedoch auf ein Referenzmuster trainiert und erkennt das vorliegende Muster nicht. Dies läßt sich vermeiden, wenn jeweils eine Ereignisspur als neues Lernmuster dient oder das erzeugte Muster als Referenzspur in der erzeugenden Station gespeichert wird.

Die Abstände zwischen den einzelnen SYNC-Rahmen haben Einfluß auf die Zuordnung. Aquidistante Abstände führen zu einer nicht eindeutigen Zuordnung, dasselbe gilt für periodisch wiederholte Abstandsmuster. Eindeutig kann die Zuordnung nur erfolgen, wenn ein zufälliges Muster zugrundeliegt.

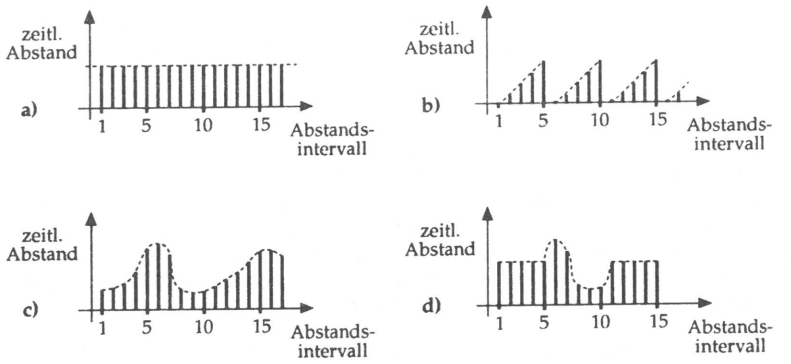


Bild 4.11: Muster für Abstandsfolgen: a) konstant, b) rampenförmig, c) zufällig, d) Kombination aus konstant und zufällig

Unabhängig von einer Referenzspur ist man, wenn statt eines Neuronalen Netzes die Korrelation angewandt wird. Der Korrelationskoeffizient c liefert ein Maß für die Ähnlichkeit von zwei diskreten Folgen F^i und F^j, die um l Abstandsintervalle gegeneinander verschoben sind:

$$c(1) = \frac{\sum_k (T_k^i - T^i)(T_{k+1}^j - T^j)}{\sqrt{\sum_k (T_k^i - T^i)^2}\sqrt{\sum_k (T_{k+1}^j - T^j)^2}} \qquad \text{(Gl.4)}$$

mit T^i, T^j als Abstandsmittelwerte der beiden betrachteten Folgen mit k Funktionswerten.

Dort, wo der Korrelationskoeffizient maximal ist, sind die Folgen am ähnlichsten und bei $c=1$ sind die Folgen identisch. Seine Grenzen hat die Zuordnung über die Korrelationsberechnung dann, wenn die zeitlichen Abstände des Musters äquidistant sind oder Folgenanteile mit geringer Periodendauer wiederholt werden.

Ähnlich der Korrelation liefert die diskrete Faltung einen Wert, der die Ähnlichkeit der Abstandsfolgen angibt. Berechnet werden kann diese entweder als direkte Faltung oder über die schnelle Faltung, wenn man die diskrete Fourier-Transformation und deren Inverse verwendet.

Zusätzliche Daten der SYNC-Rahmen (Länge oder Inhalt) können zur Kontrolle der korrekten Spurzuordnung nach der Korrelationsrechnung verwendet werden. Die Zuordnung allein über Rahmeninhalte (z.B. eine steigende Nummer im Datenteil der SYNC-Rahmen) erlaubt eine schnelle Zuordnung, jedoch sind ständig neue SYNC-Rahmeninhalte erforderlich, was eine unabhängige Erkennung der SYNC-Rahmen und Extrahierung der Nummern verlangt. Ähnliches gilt, falls die Länge der SYNC-Rahmen veränderbar ist und als Zuordnungskriterium dient. Diese Möglichkeiten führen wiederum auf Folgen, die mit den beschreibenen Verfahren bearbeitet werden können.

Zusätzlich lassen sich zwei Ansätze (SYNC-Abstände und -Inhalte) kombinbieren, indem die Zuordnung über die Zeitabstände erfolgt und mit den in den SYNC-Ereignissen erfaßten Rahmeninhaltswerten auf Korrektheit überprüft wird.

Bisher wurde angenommen, daß eine ausgezeichnete Station diese SYNC-Rahmen erzeugt. Verwendet man dabei eine vordefinierte Abstandsfolge, so vereinfacht dies die Zuordnung, da diese bekannte Referenzfolge als zusätzliche Zuordnungshilfe dienen kann. Werden dagegen die SYNC-Rahmen in willkürlichen Abstand versandt, so werden damit die Muster eindeutiger, jedoch ist keine Referenzfolge mehr vorhanden. Es muß nun entweder eine Abstandsfolge als Referenzfolge ausgewählt oder eine solche Referenzfolge aus allen Folgen erzeugt werden. Diese letzte Methode muß die Eigenschaften der lokalen frei laufenden Uhren mitbeachten, so daß jeder Zeitstempel eines SYNC-Rahmens als unsichere Angabe betrachtet werden muß.

Weitere Erschwernis bei der Zuordnung sind die möglichen Verluste von SYNC-Rahmen, die am Netz auftreten können:

- die Messung wird nicht in allen Monitorkomponenten zur selben Zeit gestartet, so daß die Abstandsfolgen bei unterschiedlichen SYNC-Ereignissen beginnen,
- die erkannten Folgen stimmen nicht mit dem Referenzmuster überein, wenn einzelne
  SYNC-Rahmen im Netz verloren gehen,
  SYNC-Rahmen im Netz verzögert werden,
  SYNC-Rahmen erst verspätet vom Absender verschickt werden,
  SYNC-Ereignisse in den Monitorkomponenten verloren gehen,
- Monitorkomponenten können nicht alle SYNC-Rahmen erkennen oder
- die Messung ist so kurz, daß nur wenige SYNC-Rahmen erkannt werden.

Diese Schwierigkeiten können einzeln oder in Kombination auftreten.

Wird nur eine sehr kurze Messung gestartet, so sind nur wenige SYNC-Ereignisse in den einzelnen Ereignisspuren enthalten, die nicht notwendigerweise zum selben Zeitpunkt begonnen haben. Solch kurze Messungen, die für den Meßsystemanwender kurze Wartezeiten zur Folge haben, verlangen eine hohe SYNC-Rahmenrate im Netz. Dies führt zu einer stärkeren Systembeeinflussung und größeren Meßfehlern. Andererseits darf die SYNC-Rahmenrate nicht zu gering sein, da sonst die Abweichung der frei laufenden lokalen Uhren nicht mehr kompensiert werden kann. Zusätzlich benötigt jedes SYNC-Ereignis auch einen Eintrag im dezentralen Meßspeicher, so daß von dieser Seite eine möglichst geringe SYNC-Rahmenrate gewünscht wird.

Versucht man, aus Rechenzeitgründen mit möglichst wenigen SYNC-Abständen eine sichere Zuordnung der einzelnen Ereignisspuren vorzunehmen, so erfordert die Berechnung der Korrelation die wenigsten Rechenschritte. Deshalb wird in diesem verteilten Meßsystem die Zuordnung der Meßspuren über die Korrelation durchgeführt.

Nachdem die zueinanderpassenden SYNC-Ereignisse gefunden sind (Bild 4.12 erster Schritt), korrigiert man die einzelnen Zeitlinien auf eine globale Zeitlinie (Bild 4.12 zweiter Schritt). Dabei

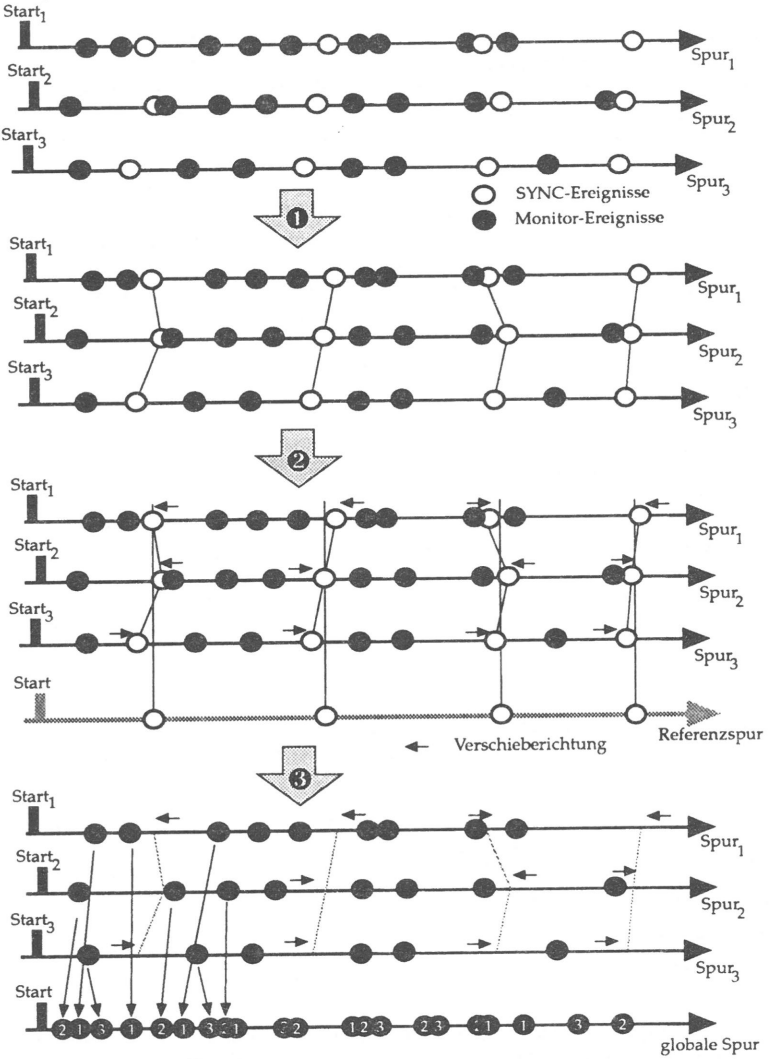


Bild 4.12: Resynchronisation:
Schritt ❶ Zuordnung der SYNC-Rahmen,
Schritt ❷ Bildung der Referenzspur aus den lokalen Spuren und
Schritt ❸ Zusammenbau der globalen Ereignisspur.

wird der Mittelwert aller sich entsprechenden SYNC-Zeitwerte als neuer Zeitwert angenommen unter Berücksichtigung der vorgegebenen Laufzeiten.

Eine anschließende Interpolation ist nötig, weil die Resynchronisation der lokalen Uhren über die Zuordnung der in der Ereignisspur eingebetteten SYNC-Ereignisse erfolgt. Der Zeitwert eines normalen Ereignisses wird anschließend auf die korrigierte Zeitlinie seiner Spur umgerechnet, wozu ein übliches Interpolationsverfahren dienen kann. Liegen die SYNC-Ereignisse zeitlich nahe beieinander, so darf linear interpoliert werden. Weitere Möglichkeiten sind Akima- oder Spline-Interpolation [2].

Abschließend werden alle SYNC-Ereignisse aus den einzelnen Ereignisspuren entfernt und diese zeitlich in die globale Ereignisspur einsortiert. Gleichzeitig findet eine Kennzeichnung der einzelnen Ereignisse statt, so daß bei der späteren Auswertung der Herkunftsort erkennbar bleibt. Darüberhinaus kann eine Anpassung einzelner Ereignisfelder erfolgen, damit die Auswertungsprogramme mit maschinengeeigneten Datenformaten arbeiten können.

4.4 Meßdatenauswertung

Die direkte Anzeige der extrahierten Informationen ist die einfachste Form der Auswertung. Jedes in einem Ereignis gespeicherte Feld wird in lesbarer Form dem Benutzer aufgelistet. Umrechnungen sind dabei nur für die Zeitstempel nötig, sofern der Benutzer anstelle des Zykluszählwertes die entsprechende Absolutzeit wünscht. Alle Ereignisse einer Spur werden in Tabellenform aufgelistet. Die Interpretation bleibt dem Benutzer überlassen. Eine Variante erlaubt es, die vom Objektsystem generierten Informationen unter, vom Benutzer vordefinierten, Namen anzuzeigen. Da alle Informationen in binärer Kodierung in den Ereignisspuren vorliegen, sind unterschiedliche Konvertierungen von der binären in darstellbare Formate sinnvoll. Basierend auf den neuen Darstellungen lassen sich dann mit Standardwerkzeugen (*awk*, *sed* und *grep* in [181]) schnell eigene Filter zur gezielteren Auswertung erstellen.

Bei der direkten Auswertung und der statistischen Auswertung ist es gleichgültig, ob eine lokale Ereignisspur oder die globale Ereignisspur ausgewertet wird. Dies ist möglich, da das interne Format einer lokalen und globalen Ereignisspur erkannt wird.

Das Ereignisformat wird durch die Speicherbelegung in den Monitorkomponenten vorgegeben. Die je Ereigniseintrag vorhandenen Felder sind:

- Informationsfeld,
- Zeitstempel und
- Meßfühleradresse.

Im Ereigniseintrag sind keine Felder für Statuskennungen des Meßfühlers oder der Monitorkomponente enthalten. Überläufe der FIFO-Speicher oder der Uhren werden nicht verzeichnet. Treten FIFO-Überläufe auf, so ist die externe Monitorkomponente zu langsam. Dieser Fall kann im Betrieb nicht auftreten, da das Meßsystem so dimensioniert wurde, daß immer jeder Informationswert sofort (oder spätestens im nächsten Abfrageintervall) übernommen wird. FIFO-Überläufe sind somit Indikatoren für Fehlverhalten des Systems. Einen Überlauf der Uhr erkennt man am Sprung der Zeitwerte von sehr großen zu sehr kleinen Zahlenwerten. Dies tritt immer dann auf,

wenn die gewählte maximale Meßperiode bei einer Messung überschritten wird. Dieser Effekt läßt sich bei der anschließenden Auswertung kompensieren. Ein Uhrenüberlauf ist somit kein Fehler.

4.4.1 Datenbeschreibungssprache

Durch die Trennung von Meßdaten (Information, Adresse und Zeitstempel) von den Beschränkungen der Meßmittel (Überläufe oder Abbruch) kann die Ereignisspur als reiner Datenkanal betrachtet werden, zu dem parallel der Befehlskanal (abgewickelt über das Kommunikationsprotokoll) besteht. Eingebettet in diesen Datenkanal ist ein transparenter Kanal, der für den Transfer der im Objektsystem generierten und extrahierten Informationen zuständig ist. Dieser Informationskanal wird um die in den Monitorkomponenten erzeugten Meßfühleradressen erweitert.

Der erweiterte Informationskanal ist Basis für eine benutzergerechte Auswertung, die mit Namen anstelle der internen Kodierungen arbeitet. Grundlage dafür ist eine Sprache, mit der die Interpretation der Kodierungen von Ereigniseinträgen vorgebbar ist. Es ist nicht Ziel dieser Beschreibungssprache, daß daraus Meßstatements oder Meßpunkte an der Hardware des Objektsystems erzeugt werden. Dieser Schritt bleibt dem Benutzer überlassen.

Jedes Informationsfeld hat eine feste Größe, die sich beliebig in Teilfelder unterteilen läßt. Damit kann der Benutzer die für ihn günstige Einteilung wählen. Dieses *Grobmuster* wird über eine Nummer identifiziert und ist das Gerüst für die weitere Beschreibung. Unterschiedliche Grobmuster legen die Struktur der später zu interpretierenden Daten fest.

In Bild 4.13 sind zwei Grobmuster mit den darin enthaltenen Teilfeldern skizziert. Durch Verfeinerung der Grobmuster erhält man *Feinmuster*, die mit einem Ereignisnamen versehen werden dürfen. Die spätere statistische Auswertung arbeitet nur noch mit diesen benannten Feinmustern.

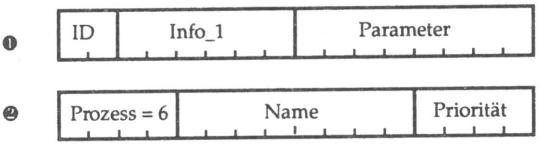


Bild 4.13: Grobmuster ❶ und ❷ mit Teilfeldern

Der Zugriff und die Beschreibung der Inhalte der Teilfelder der Grob- und Feinmuster erfolgt ebenfalls über Namensangaben. Jedes Teilfeld wird bei der Grobmusterspezifikation benannt und mit der gewählten Größe definiert. Die Lage des Teilfeldes innerhalb des gesamten Informationsfeldes ergibt sich aus seiner Position in der Reihe der Teilfeldbeschreibungen.

Das Feinmuster *ProzedurEnde* basiert auf Grobmuster ❶ mit den einzelnen Teilfeldern *ID*, *Info_1* und *Parameter*, die hier mit konkreten Wertvorgaben feiner spezifiziert wurden: *ID=0* und *Info_1=0x31*. Die Teilfeldinhalte können auf verschiedene Weise beschrieben werden:

- Inhalt unbestimmt,
- Inhalt mit fixem Wert vorbesetzt,

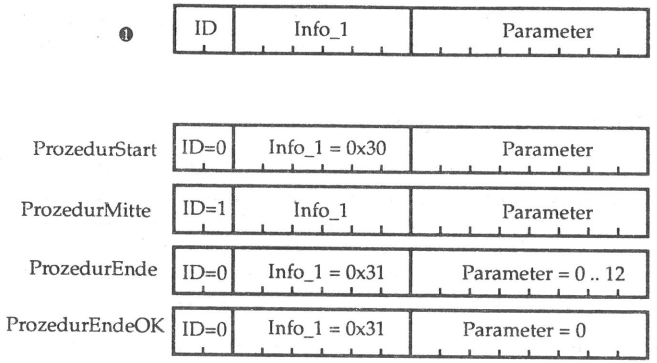


Bild 4.14: Unterschiedliche Feinmuster, basierend auf Grobmuster ❶

- Inhalt aus Wertebereich oder
- Inhalt nicht innerhalb von Wertebereich.

Bei dem Beispiel aus Bild 4.14 sind die ersten 3 Feinmuster disjunkt. Dies muß nicht notwendigerweise immer gelten. Auf diese Weise lassen sich aus Mengen weitere Teilmengen bilden, wie bei *ProzedurEnde* und *ProzedurEndeOK*.

Diese einfachen Muster reichen aus, wenn je Ereignis genau ein Eintrag in die Ereignisspur ausreicht. Sind die zu übergebenden Informationsmengen größer, so müssen mit diesem Konzept mehrere Ereigniswerte nacheinander vom externen Monitor aufgesammelt werden. Dies führt nun dazu, daß ein logischer Eintrag aus mehreren Grobmustern besteht, die unabhängig voneinander beschrieben werden müssen. Der logische Zusammenhang wird auf der Ebene der Feinmusterspezifikation bekanntgemacht, indem die Folge der zu diesem Feinmuster gehörigen Grobmuster angegeben wird.

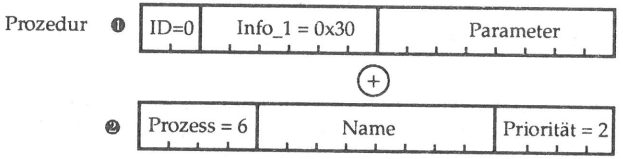


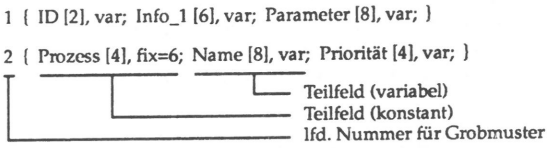
Bild 4.15: Feinmuster, aufgebaut aus zwei verketteten Grobmustern

Vorteilhaft ist hierbei, daß einfach Grobmuster verkettet und verfeinert werden können (Bild 4.15) und bei der Auswertung die Zeitstempel der einzelnen Feinmuster sichtbar bleiben.

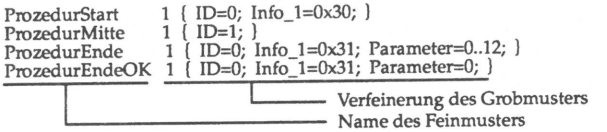
Zur maschinellen Erkennung der Ereignisse bedarf es einer Syntax, mit der diese Grob- und Feinmuster beschrieben werden. Bei der Grammatik bieten sich linksrekursive Produktionsregeln (LALR(1)) an, die per Definition widerspruchsfrei sind. Für Sprachen auf dieser Basis existieren

leistungsfähige Entwurfswerkzeuge, mit denen die Syntax- und Semantikprüfung unterstützt wird. Einzig die angestoßenen Aktionen sind problemabhängig zu entwickeln.

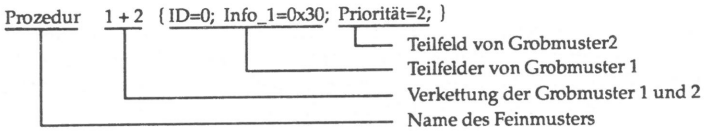
Die Beschreibung der Ereignisformate basiert auf einer Beschreibungsform, die der Definition von Strukturen in den Programmiersprachen PASCAL oder C stark ähnelt.



Die erste Zeile beschreibt Grobmuster ❶, die zweite Zeile das Grobmuster ❷ aus Bild 4.13. Der Verfeinerungsschritt entsprechend Bild 4.14 folgt mit:



und für das verkettete Grobmuster nach Bild 4.15:



Mit diesen Eingabedefinitionen kann eine Erkennung der zugehörigen Ereignisse in der Ereignisspur stattfinden. Die Erkennung der Muster und die Vorbereitung für die statistische Auswertung erfolgt über die Schritte:

- Syntaxanalyse,
- Semantikanalyse,
- Identifizierung der Ereignisse in der Ereignisspur und
- Erzeugung einer Referenzdatenbank.

Unter Verwendung dieser Datenbeschreibung kann erkannt werden, ob ein betrachtetes Ereignis zur vorgegebenen Beschreibung paßt. Jedes Ereignis kann:

- keinem Feinmuster,
- genau einem Feinmuster oder
- mehreren Feinmustern entsprechen.

Der Bearbeitungsschritt, der die Identifizierung der Ereignisse der Spur durchführt, stellt die Zuordnung zwischen der gemessenen Binärinformation aus der Ereignisspur und der namentlichen Kennung für die Benutzer her. Da diese Zuordnung keine 1:1 Zuordnung darstellen muß, sind separate Listen zu bilden. Jede Liste ist für ein spezielles Feinmuster gültig und enthält alle in der Ereignisspur gefundenen Einträge. Einträge, die mehreren Feinmustern genügen, erscheinen somit in mehreren Listen.

Um Speicherplatz zu sparen, enthält die erstellte Liste nicht die betreffenden Einträge selbst, sondern während der Zuordnung der Binärinformation an die Namen über die Datenbeschreibungssprache wird eine Referenz erstellt. Jede Liste enthält nur Verweise auf die Einträge in der Ereignisspur. Die Spur selbst wird nicht modifiziert und kann somit beliebig oft mit unterschiedlichen Datenbeschreibungen ausgewertet werden. Alle Listen mit enthaltenen Verweisen bilden zusammen die Referenzdatenbank.

Solange die Datenbeschreibung nicht geändert wird, kann die Referenzdatenbank für unterschiedliche, auch statistische, Auswertungen dienen. Erst mit der Erstellung einer neuen Beschreibung müssen die Einträge in der Ereignisspur neu zugeordnet werden, wobei dann neue Listen für die Referenzdatenbank entstehen.

4.4.2 Statistikauswertung

4.4.2.1 Statistische Größen

Im Rahmen dieser Arbeit werden Ereigniseinträge in Meßspuren betrachtet, wobei die Zeitstempel dem Abfrageintervall während der Messung entsprechen. In diesem diskreten Fall beschreibt die Verteilung A_k die Wahrscheinlichkeit eines Ereignisses (Zufallsvariable A) im k-ten Abfrageintervall. Daraus folgt die Verteilungsfunktion $A(k)$ sowie die Verteilungsdichtefunktion $a(k)$, die mit einem Dirac-Impuls δ an der Stelle i gewichtet wird. Wird mit der Zufallsvariablen A der Abstand zwischen zwei Ereignissen betrachtet, so entsteht eine diskrete Verteilung der Ereignisabstände.

In beiden Fällen wird keine Abhängigkeit zwischen den einzelnen Werten der Zufallsvariablen A berücksichtigt, so daß ein Erneuerungsprozeß mit unabhängigen Ereignisabständen und gleicher Verteilung angenommen wird.

Verteilung: $$a_k = P\{A = k\}$$

Verteilungsfunktion: $$A(k) = P\{A \le k\} = \sum_{i=0}^{k} a_i$$

Verteilungsdichtefunktion: $$a(k) = \sum_{i=0}^{k} a_i \delta(k - i)$$

Die aus einer diskreten Verteilung berechenbaren statistischen Kenngrößen sind:

Erwartungswert: $E[h(k)] = \sum_{k=0}^{\infty} h(k)a_k$

l – tes gewöhnliches Moment: $m_l = E[A^l] = \sum_{k=0}^{\infty} k^l a_k$

l – tes zentrales Moment: $\mu_l = E[(A - m_1)^l] = \sum_{k=0}^{\infty} (k - m_1)^l a_k$

Mittelwert: $E[A] = m_1$

Varianz: $VAR[A] = \mu_2$

Standardabweichung: $\sigma_a = \sqrt{VAR[A]}$

Variationskoeffizient: $c_a = \frac{\sigma_a}{m_1}$

4.4.2.2 Meßauswertung

Nach der Resynchronisation und Zuordnung über die Datenbeschreibung liefert die statistische Auswertung Resultate, die auf den gefundenen Feinmustern basieren.

Die Arbeitsweise ist dabei folgendermaßen: durch statistische Verfahren werden alle gewünschten Leistungswerte der in einer Liste der Referenzdatenbank gefundenen Ereignisse berechnet. Sinnvoll ist die Statistik für Typen (einzelne Feinmuster), Paare sowie bedingte Folgen von Feinmustern. Statistische Größen werden dabei bezogen auf den Ereignisabstand.

- Bei der *Typstatistik* werden alle Leistungsgrößen nur für ein Feinmuster berechnet. Beginn und Ende eines Intervalls werden durch gleiche Feinmuster definiert.
- Die *Paarstatistik* erlaubt es, daß zwei unterschiedliche Feinmuster als begrenzende Ereignisse definiert werden, wobei die Reihenfolge signifikant ist. Beim Paar A→B beginnt das Intervall immer mit einem Ereignis von Feinmustertyp A und endet mit einem Feinmuster von Typ B. Der Sonderfall A→A entspricht der Typstatistik.
- Komplexere Formen sind Serien von Ereignissen, die dazu dienen können, spezielle Folgen von Ereignissen in der gemessenen Spur zu finden. Diese Möglichkeiten umfassen die strikt einzuhaltenden Folgen, die zwingend eine vorgegebene Reihenfolge verlangen oder Folgen, bei denen alternative Teilfolgen enthalten sind.

Weiterverarbeitungsmöglichkeiten sind über Schnittstellen vorgesehen, die vorverarbeitete Daten in aufbereiteter Form nach Typ-, Paar- oder Seriendetektierung weiterreichen. Damit lassen sich anstelle der statistischen Auswertung Animationen erzeugen oder dynamische Veränderungen, wie Einschwingvorgänge, während der Meßphase aufzeigen.

Sind die Meßwerte durch Statistikberechnungen zu Resultatwerten komprimiert, so existiert noch die Schnittstelle zu weiteren Programmen, die etwa aus Einzelergebnissen Diagramme für Parameterstudien bilden oder die gemessenen Klassenverteilungen approximativ an Verteilungsfunktionen anpassen [13, 35, 178].

5 Realisierung der Meßteile

Als Umgebung für die prototypische Realisierung wurde ein am Institut verlegtes Lokales Netz, basierend auf den CSMA/CD-Medienzugriffsverfahren, ausgewählt. Der auf diesem Netz meßbare Verkehr entspricht den in [61] gemessenen Verkehren einer Entwicklungsumgebung.

Als Objektsysteme standen zwei Hardware-Plattformen zur Verfügung. Die Rechner SIEMENS WS20 [188] und Intel310 [95] basieren auf demselben Standardprozessor [91], jedoch sind die Systembusse unterschiedlich ausgeführt. Bei der WS20 wird eine modifizierte Version des genormten IBM AT-Systembus [87] verwendet. Dieser Bus beinhaltet die nötigen Signalleitungen, um neben Speicherzugriffen auch Ein-/Ausgabezugriffe abzuwickeln. Die dann ablaufenden Buszyklen werden über zugeordnete Maschinenbefehle angestoßen. Der Rechner Intel310 ist ein konfigurierbares System auf Basis des Multibus I, der spezielle Busse für Speichererkarten, Ein-/Ausgabekarten und den eigentlichen Systembus einschließt [86, 109].

Auf diesen Plattformen sind folgende Betriebssysteme im Einsatz:

- AT/RTX86 (Echtzeit-Multitasking-Betriebssystem, lauffähig im Real Mode) [172],
- RTX286 (Echtzeit-Multitasking-Betriebssystem, lauffähig im Protected Mode) [173],
- iRMX86 (Echtzeit-Multitasking-Betriebssystem, lauffähig im Real Mode) [93],
- iRMX II (Echtzeit-Multitasking-Betriebssystem, lauffähig im Protected Mode) [94],
- XENIX V (Multiuser-Multitasking-Betriebssystem, lauffähig im Protected Mode) [181].

Tabelle 11 zeigt, welche Schnittstellen auf den jeweiligen Rechnersystemen vorhanden sind.

	WS20	Intel310
AT/RTX86	IHI	
RTX286	IHI	
iRMX86		MIP
iRMX II		MIP/IHI
XENIX V	IHI	

Tabelle 11: Kommunikationsschnittstellen bei verschiedenen Betriebssystemen

Bei all diesen Kombinationen ist eine Netzwerksoftware verwendbar, die alle Schichten bis zur Transportschicht umfaßt [88]. Der Zugriff auf die gebotenen Dienste wird von unterschiedlichen Schnittstellen bereitgestellt. Beide Schnittstellen (IHI [58, 185] und MIP [88]) stellen nur einen Transfermechanismus zur Übergabe von Auftragsblöcken (engl. Request Blocks) bereit. Der Aufbau der Requestblöcke ist durch den Einsatz der iNA 960 [88, 89] als unterlagertes Transportsystem bei allen Systemen identisch. Netzzugangsbaugruppe ist hier die CP536* [186], die zur Abarbeitung der Protokollschichten 2 bis 4 über einen eigenen Prozessor verfügt. Dieser erhält seine Aufträge über die Requestblöcke und steuert den für die Abwicklung des Medienzugriffsprotokolls

eingesetzten LAN-Koprozessor [92]. Der Anschluß ans Netz erfolgt über ein AUI-Kabel (engl. Attachment Unit Interface) (10Base5) oder direkt an das Koaxialkabel (10Base2).

Von den im Meßkonzept vorgeschlagenen drei Meßstellen

- Systembus des Hostsystems (WS20-Bus, IBM AT-Bus, Multibus I),
- Bus der Netzzugangsbaugruppe [186] sowie
- Netzzugang (AUI-Kabel, LAN-Koprozessor)

wurden nur Sensoren für Hostsystembusse und den Netzzugang implementiert. Der Bus der Netzzugangsbaugruppe CP536* ist zwar bekannt, jedoch wurden die auf dieser Baugruppe installierten Programme mit residentem Betriebssystem iRMX86 vom Hersteller in einem Festwertspeicher (engl. Read Only Memory, ROM) abgelegt und sind somit nicht änderbar.

Dies führt dazu, daß die internen Abläufe der Kommunikationsprotokolle der Schichten 2b bis 4 nicht meßbar sind, sofern das Kommunikationsprotokoll auf Diensten der ISO/OSI-Schicht 4 aufsetzt. Die zur Verfügung stehende Netzzugangsbaugruppe weist keine Funktionalität in der Schicht 3 auf, so daß die Schicht 4 über diese inaktive Netzwerkschicht direkt auf den Diensten der Schicht 2 aufbaut. Die von dieser Netzzugangsbaugruppe angebotene Klasse ISO Class 4 überträgt alle Dienstdateneinheiten (engl. Service Data Units, SDU) mit Hilfe von Datagrammen der Schicht 2b. Üblicherweise werden bei der Kommunikation über Schicht 4-Dienste bei diesem Objektsystem alle SDUs auf ein Datagramm der Schicht 2b abgebildet, so daß der von den Anwendungen stammende Datenfluß erhalten bleibt. Lediglich bei Verbindungsaufbau und -abbau sowie Quittierung finden Schicht 4-interne Aktionen statt, die an der oberen Schichtengrenze nicht sichtbar sind. Darüberhinaus findet bei allen bestehenden Schicht 4-Verbindungen während Transferpausen eine periodische Überprüfung der Partnerstation (engl. Inactivity Control) statt. Auch diese Aktivitäten lassen sich bei der eingesetzten Netzzugangsbaugruppe nur am direkten Zugang auf das physikalische Netz beobachten. Segmentierung wird beim aktuellen Implementierungsstand der Schnittstelle IHI nicht durchgeführt, da der Datenbereich je übergebenem Requestblock immer kleiner als ein physikalischer Rahmen sein muß.

Die innerhalb der verbindungslosen ISO/OSI-Schicht 2b ausgeführten Bearbeitungsschritte umfassen beim Senden die Übernahme eines Paketes an einem Quelldienstzugangspunkt (engl. Source Link Layer Service Access Point, SSAP) und Weitergabe an die versendende Medienzugangsschicht (engl. Media Access Control Layer, MAC) bzw. beim Empfang vom Netz die Zuordnung der einzelnen Datagramme zum Zieldienstzugangspunkt (engl. Destination Link Layer Service Access Point, DSAP). Die softwaremäßige Ansteuerung des LAN-Koprozessors, der selbständig das Medienzugangsprotokoll abwickelt, ist Aufgabe der MAC-Schicht.

Ein direkter Zugrif auf den LAN-Koprozessor ist beim vorliegenden Transportsystem iNA 960 nicht erlaubt. Dienstzugangspunkte sind nur zur Schicht 4 (ISO TP4) und Schicht 2b (LLC Class I) vorhanden.

Für die Kommunikation über nicht-OSI konforme Protokolle (der Erweiterung von LLC Class I zum sicheren Protokoll (vergl. Bild 4.9) oder den TCP/IP-Kommunikationsprotokollen, vergl. Kap. 7.1.3) erfolgt, unter Umgehung der Schichten 3 und 4, der Zugang zum Netz zwangsweise über die ISO/OSI-Schicht 2b.

5.1 Meßfühlerentwicklungen

Mit den im folgenden beschriebenen Meßfühlern direkt am Netzzugangspunkt lassen sich die für dieses Objektsystem bestimmte Rahmen verfolgen. Die Schnittstelle zum Objektsystem selbst wird durch eine Betriebssystemerweiterung in Form eines Treiberprogrammes innerhalb der Betriebssystem-Software realisiert, so daß diese Stelle durch einen Meßfühler am Systembus beobachtet werden kann.

5.1.1 Meßfühler am Netzzugang

Die Meßfühler am Netzzugang sind Hardware-Meßfühler. Damit beschränken sich diese Meßfühler auf die Beobachtung einiger Signalleitungen am Netzzugang.

Zwei Arten von Meßfühlern an diesem Punkt wurden entwickelt:

- Meßfühler, die Aktionen des LAN-Koprozessors mitverfolgen und
- Meßfühler direkt am Netzkabel.

5.1.1.1 Meßfühler am LAN-Koprozessor

Beim Entwurf dieser ersten Meßfühler wurde entschieden, das erfolgreiche Ende eines Sende- bzw. Empfangsvorganges anhand der LAN-Koprozessormeldungen zu erkennen. Da der in der Netzzugangskarte eingesetzte LAN-Koprozessor [92] den Typen b) und d) aus Bild 4.4 entspricht, müssen deshalb die für das verteilte Meßsystem wichtigen Informationen durch Mitbeobachtung des Registerabbildes im Speicher extrahiert werden. Dieser Vorgang ist nicht trivial, da dieser Koprozessor die Registerinhalte nicht unter fixen Adressen im Speicher ablegt. Vielmehr existieren verkettete dynamische Listen: eine Liste mit Aufträgen zum Rahmensenden und eine Liste mit leeren Puffern für zukünftig zu empfangende Rahmen (Bild 5.1). Jeder Sendeauftrag enthält die vollständige Information aus Rahmendaten und Adressen sowie ein Statusfeld zur Rückmeldung. Ähnliche Verwaltungsblöcke existieren in der Empfangsliste.

Wird ein Auftrag in die Auftragsliste geschrieben, so stößt der Prozessor über eine Steuerleitung den LAN-Koprozessor an. Dieser arbeitet die Auftragsliste ab, solange Aufträge vorhanden sind. Für jeden Auftrag wird eine Rückmeldung (erfolgreich oder entsprechende Fehlermeldung) abschließend in das Statusfeld des Auftragsverwaltungsblocks geschrieben. Zusätzlich kann je Auftrag eine Unterbrechung (engl. Interrupt) an den Prozessor gemeldet werden.

Damit der LAN-Koprozessor Rahmen vom Netz empfangen kann, muß leerer Pufferraum in der Empfangsliste verfügbar sein. Solange ein Puffer noch nicht voll ist, werden die eintreffenden Rahmendaten dort gespeichert. Wird das Pufferende erreicht, so sucht sich der LAN-Koprozessor selbständig den nächsten freien Puffer und füllt auch diesen. Die Zuordnung von einzelnen Puffern zu einem Rahmen wird vom LAN-Koprozessor in Verwaltungsblöcken eingetragen.

Intern ist der LAN-Koprozessor in zwei separat arbeitende Einheiten zum Senden und Empfangen unterteilt. Da am seriellen Kanal nicht gleichzeitig gesendet und empfangen wird, serialisiert der Koprozessor die Aktionen. Zusätzlich verfügt dieser Baustein über ein Businterface, das die Zugriffe auf die Listen per direktem Speicherzugriff ausführt. Bestandteil des Businterface ist ein FIFO-Speicher, so daß die notwendigen Speicherzugriffe verzögert und blockweise erfolgen.

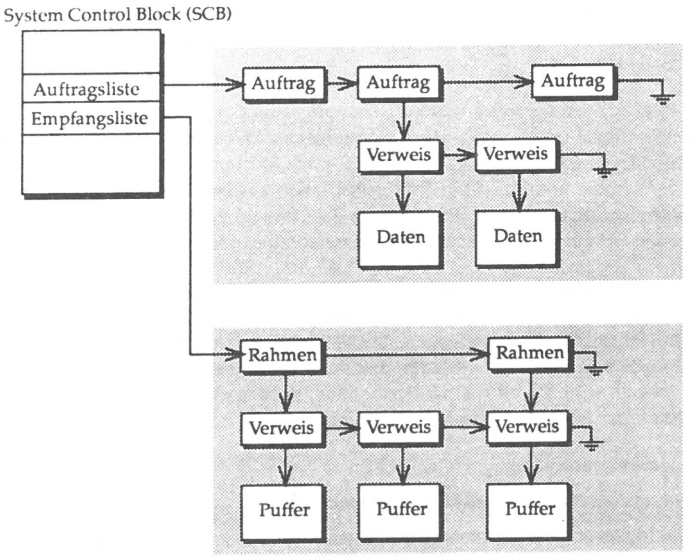


Bild 5.1: Kommunikationslisten zwischen Prozessor und LAN-Koprozessor

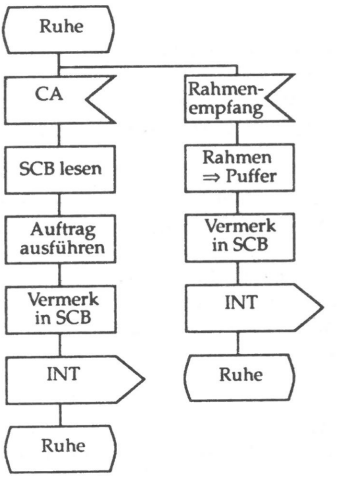


Bild 5.2: SDL-Diagramm zur Arbeitsweise des LAN-Koprozessors

Wie in Bild 5.2 gezeigt, verläßt der LAN-Koprozessor seinen Ruhezustand nur, wenn entweder durch das CA-Signal (Channel Attention) ein neuer Auftrag signalisiert wird oder Rahmendaten vom Netzkabel eintreffen.

Jeder eintreffende Rahmen wird nach vollständigem Empfang und Vermerk im SCB mit einem Unterbrechungssignal (INT) an den Prozessor gemeldet. Dasselbe geschieht, wenn einzelne Aufträge abgearbeitet werden. Während der LAN-Koprozessor noch die Auftragsliste abarbeitet, können weitere Aufträge in die Auftragsliste geschrieben werden. Da nicht feststellbar ist, ob der LAN-Koprozessor den letzten Auftrag bereits gelesen und damit das Listenende erkannt hatte, wird immer mit dem Signal CA der LAN-Koprozessor über neue anstehende Aufträge informiert.

Für den Meßfühler bedeutet dies, daß mit den Signalen CA und INT wichtige Zustände erkannt werden können. Ob der LAN-Koprozessor Netzdaten empfangen hat, läßt sich aus den Steuerleitungen zum Netzanschlußkabel ableiten. Die Prüfung, welcher Befehl erteilt wurde und ob die Aktion erfolgreich terminierte, verlangt, daß der Zugriff auf die Statusfelder im SCB und die Listen mitverfolgt wird. Damit sind als Beobachtungsstellen dieser Netzzugangsmeßfühler folgende Signalleitungen des LAN-Koprozessors wichtig:

- Synchronisationsleitungen (CA, INT),
- Leitung zum Netzanschluß (Carrier Sende, CRS),
- gemultiplexter Adreß-/Datenbus (AD0..AD15),
- Schreib-/Lesesteuerleitungen (S0, S1),
- Taktleitung (CLK) und
- Rücksetzleitung (RESET).

Die Netzanschlußleitung CRS (Carrier Sense) signalisiert den Empfang von Netzdaten. Selbst beim eigenen Senden wird CRS aktiv. Mit den S0/S1-Leitungen zeigt der LAN-Koprozessor, welche Speicherzugriffsart (Lesen/Schreiben) gerade stattfindet. Eine genauere Analyse zeigt, daß vom gemultiplexten Adreß-/Datenbus die Leitungen AD15-AD13, AD8 sowie AD2-AD0 hier ausreichend sind.

Grundlage des Meßfühlers ist ein endlicher Automat (Bild 5.3), der durch die oben genannten Eingangssignale alle relevanten Zustände des LAN-Koprozessors nachverfolgen kann.

In Bild 5.3 sind die Zustände *Z14* und *Z50* markiert, da diese Zustände nach Auftreten des CA-Signals erreicht werden. Zustand *Z49* aktiviert das INT-Signal, das mit dem CA-Signal in *Z50* quittiert wird und bei *Z51* inaktiv gesetzt wird. Das abgebildete Automatenmodell listet nur einige wenige Befehle für den LAN-Koprozessor auf:

NOP	no operation,
TRANSMIT	Sendebefehl und
SETADDR	laden der eindeutigen MAC-Adresse in den Koprozessor.

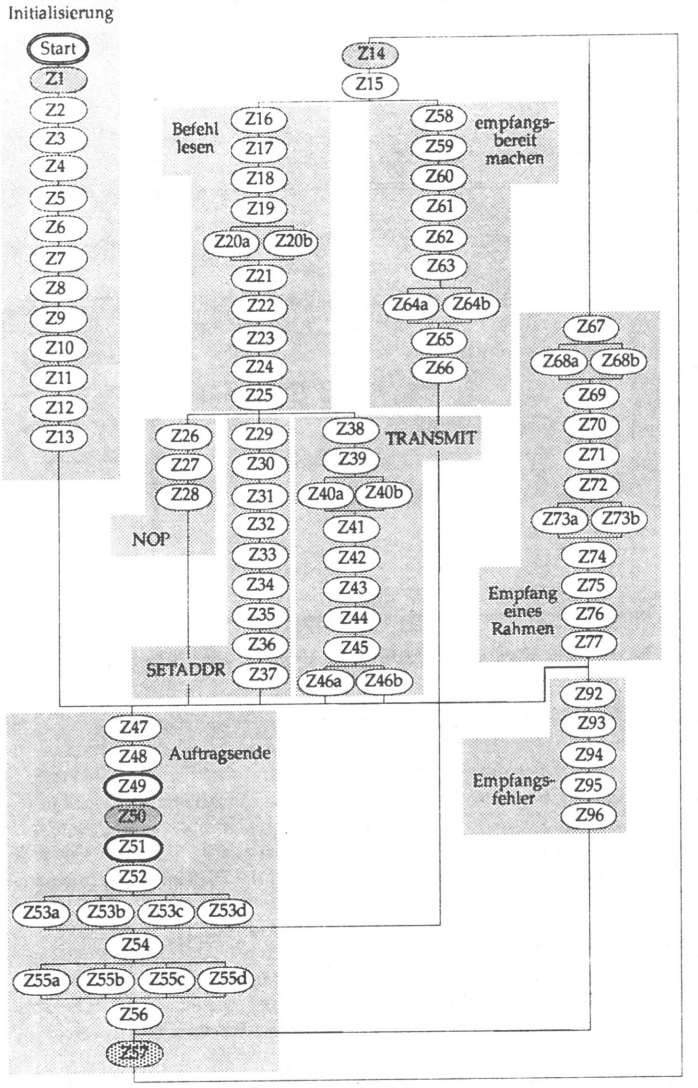


Bild 5.3: Vereinfachtes Automatenmodell

Weitere wichtige Befehle (vergl. [92]) werden über Zustandsfolgen zwischen *Z25* und *Z47* erkannt. In diesem Bild sind bis auf die Zustandsfolge beim Erkennen eines CRC-Empfangsfehlers

(*Z92...Z96*) keinerlei Fehlerfolgen eingetragen. Weiterhin sind alle Zustände, die Puffer- und Listenverwaltung betreffen, in Bild 5.3 weggelassen. Das vollständige Automatenmodell ist in [54] beschrieben.

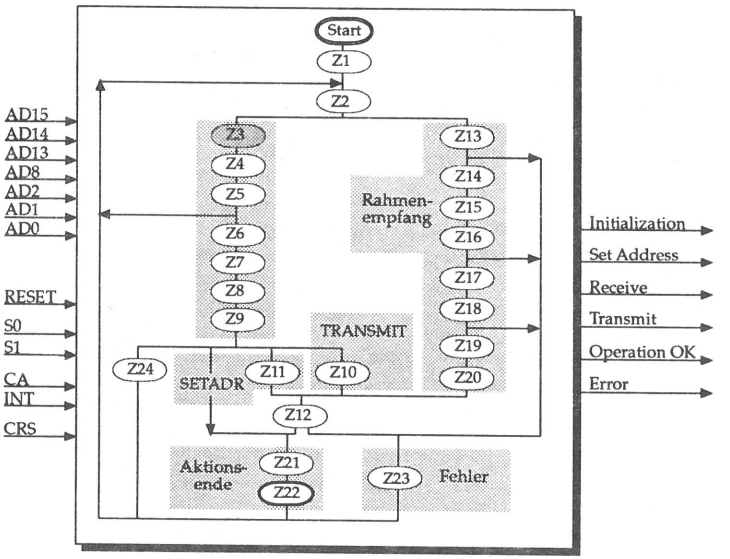


Bild 5.4: Realisierter Meßfühler (reduzierte Version) am LAN-Koprozessor

Für eine Realisierung mit am Institut vorhandenen Werkzeugen war dieser Zustandsraum viel zu groß. Durch eine gezielte Reduzierung der Zustandsanzahl wurden im Leistungsumfang reduzierte Varianten des großen Automaten erstellt. Diese kleineren Automaten konnten dann mit programmierbaren PAL-Bausteinen (engl. Programmable Array Logic) realisiert werden. Ein realisierter Meßfühler ist in Bild 5.4 skizziert. Die mit diesem reduzierten Meßfühler erzeugten Signale zeigen die wichtigsten Aktionen sowie deren Endestatus an. Daneben existieren weitere Versionen, so ein Automat zur Anzeige von *Transmit* und *Receive* mit nur noch 19 Zuständen, der in einem einzigen PAL realisiert ist.

Probleme bereitet dieser Intel-Baustein und damit auch der zu beobachtende Automat in dem Fall, wo Gruppenadressierung verwendet wird. Der LAN-Koprozessor kann keine vollständige Filterung bei dieser Adressierungsart durchführen, weshalb Rahmen, die nicht für dieses Objektsystem bestimmt sind, trotzdem empfangen und vom Meßfühler angezeigt werden.

Eine weitere Eigenschaft dieses Meßfühlers ist es, daß die Ereignisanzeige erst nach dem abschließenden Zugriff auf den System Control Block durch den LAN-Koprozessor erfolgen kann. Damit ist aufgrund der zeitlich verzögerten Speicherzugriffe des LAN-Koprozessors keine feste zeitliche Zuordnung mit der durchgeführten Aktion mehr herstellbar. Anders als in [22] definiert hier die Koprozessor-interne Bearbeitung den Meßpunkt, vergl. Bild 5.5.

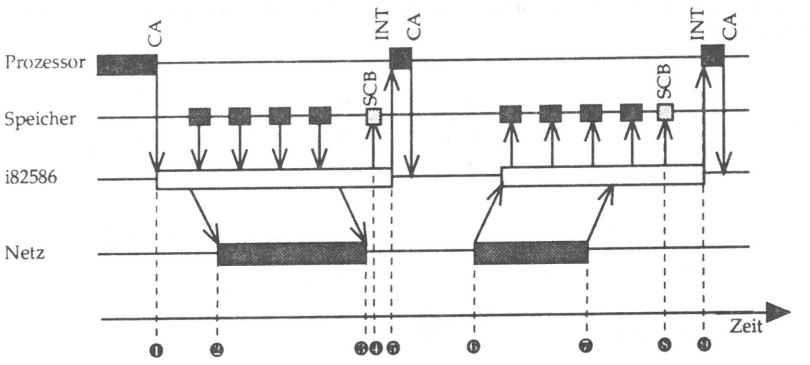


Bild 5.5: Zeitverhalten am Meßfühler

Im Falle eines Sendeversuchs (❶ bis ❺) nach Bild 5.5 durchläuft der Automat seine Zustände, bis er zum Zeitpunkt ❹ erkennt, ob der Transfer erfolgreich war oder nicht. Einen Rahmenempfang (❻ bis ❾) erkennt der Automat durch aktives CRS ohne vorangehendes CA. Beim Zugriff auf den SCB (❽) wird wieder entschieden, welche Meldung zu generieren ist.

5.1.1.2 Meßfühler direkt am Netzzugang

Mit dem oben beschriebenen Meßfühler am LAN-Koprozessor wird die Netzzugangsschnittstelle mitbetrachtet. Bei dieser Schnittstelle gibt es nur sehr wenige Leitungen, so daß sich dieser Punkt zur weiteren Beobachtung anbietet. Gleichzeitig ist dies der Punkt, an dem die Rahmendaten das Objektsystem verlassen.

Hier wird als Meßpunkt die Verbindung zwischen dem LAN-Koprozessor und dem Kodier-/Dekodierbaustein (engl. Encoder/Decoder) gewählt. Alle Signale an dieser Schnittstelle (Tabelle 12) sind Standard-TTL Signale mit NRZ-Kodierung, die dann im Encoder/Decoder in Manchesterkodierte differentielle Signale entsprechend der Norm ISO 8802-3 [99] gewandelt werden.

Signal	Bedeutung
RxD	serielle Empfangsdaten (engl. Receive Data)
RxC	Empfangstakt (engl. Receive Clock)
CRS	Empfangsanzeige (engl. Carrier Sense)
COL	Kollisionsanzeige (engl. Collision)

Tabelle 12: Signalleitungen zwischen LAN-Koprozessor und Enkoder/Dekoder

Zweck eines Meßfühlers an diesem Meßpunkt ist es, beliebige Muster in den transferierten Rahmen zu erkennen. Dazu werden die seriellen Datenströme analysiert. Aufgrund des an dieser Stelle noch sichtbaren MAC-Protokolls (CSMA/CD) kann es zu Kollisionen kommen, die vom Meßfühler anhand des COL-Signals erkannt werden.

Die Meßmöglichkeiten, die mit unterschiedlichen Meßfühlern an diesem Meßpunkt realisiert sind, zeigt Bild 5.6.

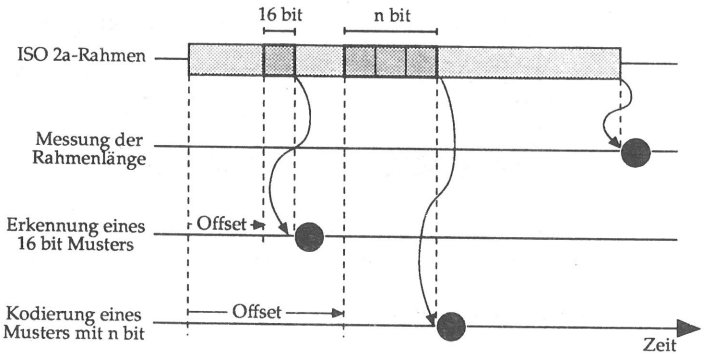


Bild 5.6: Meßmöglichkeiten mit rückgekoppeltem Schieberegister

Zum Einsatz kommt hierbei ein *rückgekoppeltes Schieberegister* (RSR). Durch unterschiedliche externe Ansteuerung wird einer der drei in Bild 5.6 gezeigten Betriebsmodi gewählt. Dasselbe RSR wird auch zur Bestimmung der Verzögerung (engl. Offset), d.h. zum Abzählen der seriellen Datenbits bis zum Musterbeginn, eingesetzt. Im Zählmodus wird vor jedem Meßschritt der entsprechende Vorgabewert in das RSR geladen und mit dem Takt der seriellen Daten um eine Stelle geschoben. Das Zählende ist erreicht, wenn eine speziell gewählte Kennung (logische Null) im RSR steht. In diesem Zustand wechselt das RSR seine Betriebsart in einen der drei Meßmodi. Durch die Verwendung des RSR als Zähler entfällt ein separater Rückwärtszähler.

Beim Betrieb als Zähler und im Kodiermodus muß ein Rückkoppelmuster gewählt werden. Hierzu bieten sich primitive Generatorpolynome 16.ter Ordnung an. Bei diesen Generatorpolynomen ergibt sich eine maximale Zykluslänge mit genau (2^{16}-1) Zuständen. Der Zustand mit der binären Kodierung *0..01* (logische NULL) wird als Endzustand gewählt und darf somit innerhalb des Zykluses nicht auftreten.

Die einfachste Messung ist die Rahmenlängenmessung. Alle Bits eines transferierten Rahmens werden durch die Taktflanke (Signal RxC) für gültig erklärt, so daß nur die Taktflanken gezählt werden müssen. Nach Abzählen des Offsets wird in das RSR ein Ladewert geschrieben, bei dem zum aktuellen Generatorpolynom sichergestellt ist, daß die logische Null niemals vor dem Rahmenende erreicht wird. Gemäß der Norm ISO 8802-3 kann ein Rahmen 72...1526 octetts (Rahmen inklusive Präambel) lang sein. Da die Rahmenlänge in Bits gemessen wird, sind Werte im Bereich von 576...12208 möglich und deutlich kleiner als die maximale Zykluslänge. Der Ergebniswert bei Rahmenende stellt den momentanen Zykluszustand dar. Durch Rückrechnung auf den Ladewert kann daraus die Rahmenlänge ermittelt werden.

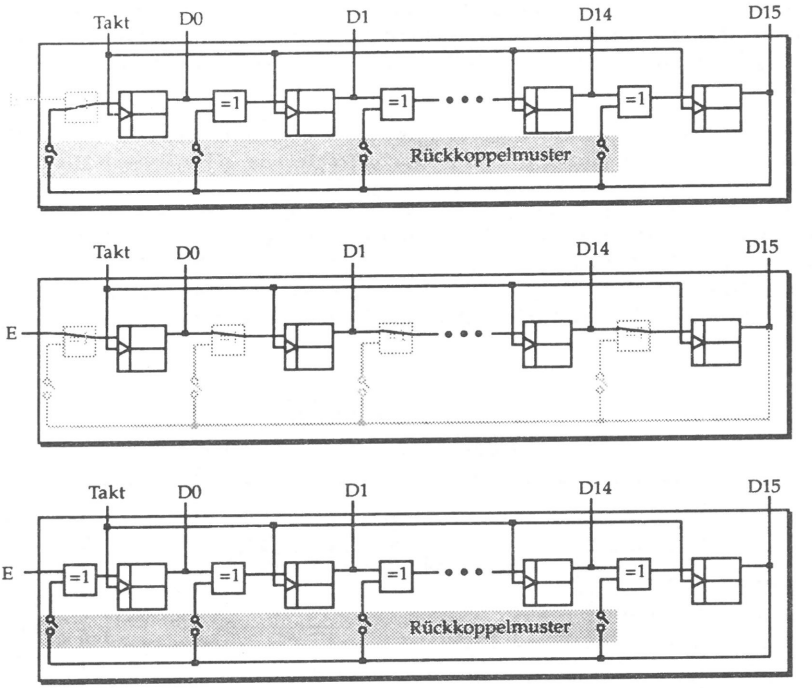


Bild 5.7: Lineares rückgekoppeltes Schieberegister: Zählmodus (oben), Transparentmodus (mittleres Bild), Kodiermodus (unten)

Für die Messung im Transparentmodus werden nach Abzählen des Offsets das RSR gelöscht (Binärmuster *0..00*) und 16 bit in die Registerkette geschoben. Rückkopplungen finden hierbei nicht statt, da die Beschaltung in diesem Modus dem aus Bild 5.7 (Mitte) entspricht. Abschließend wird der gefundene 16 bit Wert parallel an den Ausgängen D0..D15 übergeben.

Ist man an Mustern interessiert, die größer als 16 bit sind, so lassen sich mit dem RSR gemäß Bild 5.7 (unten) längere Musterfolgen aus den Netzrahmen auf 16 bit große Informationswerte abbilden. In diesem Kodiermodus wird als Musterlänge ein Vielfaches der Wortbreite (16 bit) erlaubt:

n = (16 bit, 32 bit, 48bit, 64 bit).

Wieder kann über den Zählmodus ein beliebiger Bit-Offset eingestellt werden, ab dem das n bit lange Muster erwartet wird. Mit Erreichen des Offset-Ende werden wieder das RSR gelöscht und die eintreffenden Rahmenbits durch das RSR geschoben. Über die Rückkoppelzweige und die Antivalenzgatter wird eine Abbildung der Rahmenbits auf einen 16 bit Informationswert durchgeführt.

Sind die möglichen n bit Muster a priori bekannt, so läßt sich ein Rückkoppelmuster angeben, bei dem alle eingeschobenen Muster paarweise disjunkte Informationswerte erzeugen. Dies verlangt,

daß weniger als 2^{16} Eingangsmuster unterschieden werden sollen. Die übergebenen kodierten Informationswerte sind aber zu den möglichen Rahmenmustern eindeutig zuordbar. Wichtigste Anwendung dieses Kodiermodus ist die Abbildung realer 48 bit MAC-Adressen in eindeutige 16 bit Informationswerte.

Ein Variante dieses RSR wurde in Zusammenarbeit mit dem Institut für Mikroelektronik (IMS) der Universität Stuttgart auf Basis eines Gate Array (2µ Technologie) in Form einer VLSI-Schaltung (engl. Very Large Scale Integration Circuit) gefertigt [206]. Dieser VLSI-Baustein enthält ein RSR mit zugehörigen Registern für Offset-Werte und ein 48 bit Referenzmuster (Bild 5.8). Nach Abzählen des Offsets (RSR mit 9 bit) wird ein 48 bit langes Rahmenmuster in ein Schieberegister geschoben. Nach jedem Schiebeschritt wird der aktuelle Registerinhalt mit dem Referenzwert verglichen und bei Gleichheit ein Signal generiert. Unter Verwendung externer Anpaßlogik an die AUI-Signale kann dieser Mustervergleich für jeden über den Kanal übertragenen Rahmen erneut durchgeführt werden.

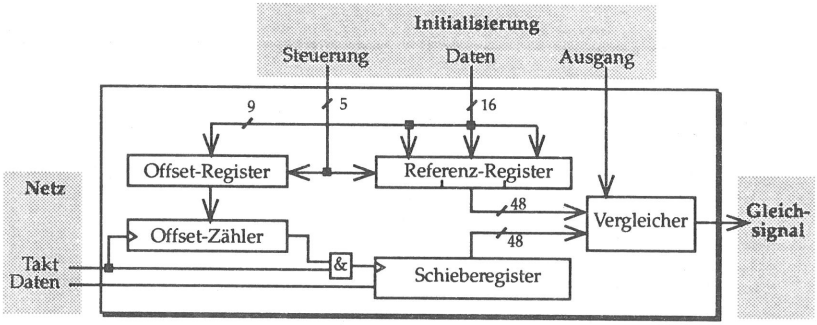


Bild 5.8: VLSI-Baustein zur 48 bit Mustererkennung (aus [206])

Bei jedem der Meßfühlervarianten (diskret aufgebautes RSR bzw. VLSI-Chip) bleibt das Ergebnis bis zum Rahmenende erhalten, sofern keine Kollision stattgefunden hat. Über eine kombinatorische Logik wird dieser Zeitpunkt erkannt und die Übergabe an die externe Monitorkomponente initiiert.

Durch Kombination der einzelnen Meßfühler lassen sich mächtigere Meßfühler bereitstellen. Auf einer Baugruppe realisiert sind:

- Sende-/Empfangserkennung,
- 48 bit Mustervergleich mit Hilfe des VLSI-Chips,
- Rückgekoppeltes Schieberegister mit einem der drei Modi:
  - Rahmenlängenmessung oder
  - 16 bit Transparentmessung oder
  - n bit Kodiermessung.

Damit sind beispielsweise folgende rahmenbezogene Messungen am Netzzugang durchführbar:

- Rahmenlänge je erfolgreich gesendetem/empfangenem Rahmen,
- DSAP/SSAP-Paar je erfolgreich gesendetem/empfangenem Rahmen,
- MAC-Zieladresse je erfolgreich gesendetem bzw. MAC-Quelladresse je erfolgreich empfangenem Rahmen,
- MAC-Zieladresse und DSAP/SSAP-Paar je erfolgreich gesendetem bzw. empfangenem Rahmen,
- alle Rahmen mit Referenzmuster sowie deren Rahmenlänge,
- alle Rahmen mit Referenzmuster sowie deren DSAP/SSAP-Paar oder
- alle Rahmen mit Referenzmuster sowie eine kodierte MAC-Adresse.

Prinzipiell möglich ist auch die Kombination von 48 bit Mustererkennug und Sende-/Empfangserkennung. Bei der aufgebauten Baugruppe [169, 50] wird jedoch bei der 48 bit Mustererkennung sofort am Rahmenende ein Informationswert an die externe Monitorkomponente übergeben, so daß der Indikator für erfolgreichen Transfer noch nicht vorliegen muß.

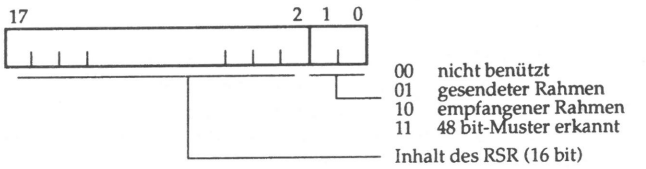


Bild 5.9: Format eines Ereigniseintrags des Meßfühlers am Netzzugang

Anhand der gespeicherten Ereigniseinträge entsprechend Bild 5.9 kann nur erkannt werden, ob der VLSI-Chip oder der Automat zur Transfererkennung die Informationsübergabe erzwungen hat. Das RSR liefert einen 16 bit Ergebniswert, der bei der späteren Auswertung in der richtigen Weise ausgewertet werden muß. Da bei der Initialisierung des RSR der entsprechende Modus gewählt wurde, ist die Betriebsart somit in der zentralen Auswertestation bekannt.

5.1.2 Meßfühler am Systembus

Alle Meßfühler für die eingesetzten Objektrechner verlangen, daß mittels Speicherzugriffen in die Monitorregister geschrieben wird.

Im System Intel310 muß dieser Schreibzugriff über den langsamen Multibus I (Zykluszeit ca. 1 µs) stattfinden [180].

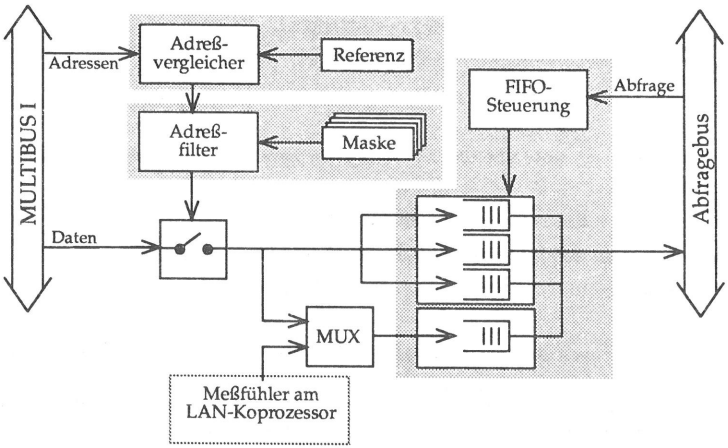


Bild 5.10: Meßfühler am Systembus Multibus I

Der in Bild 5.10 gezeigt Adreßfilter wertet einen Teil der am Systembus sichtbaren Adresse aus. Über die zugeordnete Maske kann dann eine Selektion stattfinden. Die 8bit breite Maske erlaubt es, daß 8 einzelne Wortadressen, beginnend bei der Meßfühleradresse *A*, unterscheidbar sind. Nur Informationen mit unmaskierten Adressen werden in den FIFO-Speicher geschrieben. Einzelne Meßfühler werden dabei mit Lücken *L* in aufeinanderfolgende Speicherbereiche eingeblendet (siehe Bild 5.11).

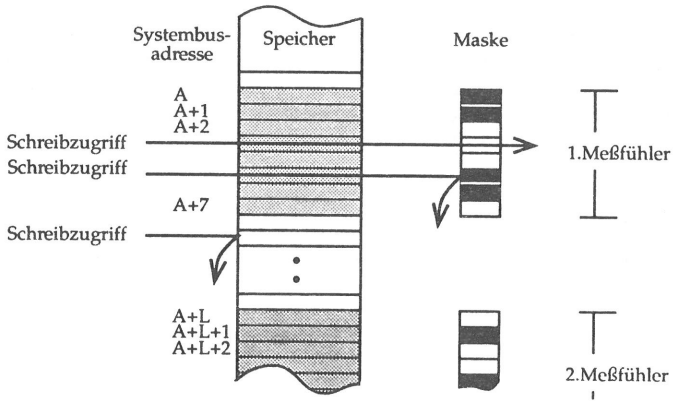


Bild 5.11: Zuordnung von Systembusadresse und Maske

Die WS20 unterscheidet am Systembus zwischen Speicher- und Ein-/Ausgabezugriffen, wobei der Speicherzugriff mit 330 ns Zykluszeit verwendet wurde. Für diesen Bus wurden drei unterschiedliche Baugruppen entwickelt:

- ❶ Version ohne Adreßfilter und nur ein FIFO (16bit breit) [62],
- ❷ Version mit Adreßfilter, 4 FIFOs (16bit breit) und Anschluß des Meßfühlers vom LAN-Koprozessor (reduzierte Version mit 4 bit Informationsbreite) [5] und
- ❸ Version mit Adreßfilter, 4 FIFOs und Anschluß des Meßfühlers vom LAN-Koprozessor (Vollversion mit 18 bit Informationsbreite).

Die Versionen mit Adreßfilter (❷ und ❸) sind sowohl im IBM-AT als in der WS20 zu betreiben. Weitere Eigenschaften der Systembus-Meßfühler sind in Tabelle 13 aufgelistet.

Zum einfachen Einbau der unterschiedlichsten Meßfühler in die Objektsysteme wurden ein bzw. vier Meßfühler in einer Baugruppe zusammen mit einer Entkopplungsstufe realisiert. Die erzeugten Meßinformationen der einzelnen Meßfühler werden in zugeordnete FIFO-Speicher eingeschrieben. Das Auslesen erfolgt durch die externe Monitorkomponente im Abfragebetrieb. Bezüglich der geschriebenen Information (16 bit) gibt es keinerlei Einschränkungen.

	❶	❷	❸
Busanschluß	WS20	WS20 / IBM-AT	WS20 / IBM-AT
ladbare Adreßreferenz		•	•
Adreßfilterung		•	•
ladbare Masken		•	•
integrierte Meßfühler	1	4	4
Breite des FIFO-Speichers	16 bit	16 bit	18 bit
Anschluß für Meßfühler LAN-Koprozessor		•	•
Infobreite für Meßfühler LAN-Koprozessor		4 bit	18 bit

Tabelle 13: Unterschiedliche Systembus-Meßfühler

5.1.3 Integration aller Meßfühler

Der Anschluß der Meßfühler am LAN-Koprozessor ist nur über einen Systembus-Meßfühler möglich. Dazu sind die Meßfühler ❷ und ❸ aus Tabelle 13 in einen transparenten Modus versetzbar, so daß die Initialisierung der Register für Generatorpolynome, Offset und Muster von der externen Monitorkomponente aus erfolgen kann, vergl. Bild 5.12.

Die spätere Unterscheidung, woher die Ereigniseinträge stammen, erfolgt durch die Meßfühleradresse, die auch zur Abfrage verwendet wird. Diese Adresse ist für eine Monitorkomponente eindeutig.

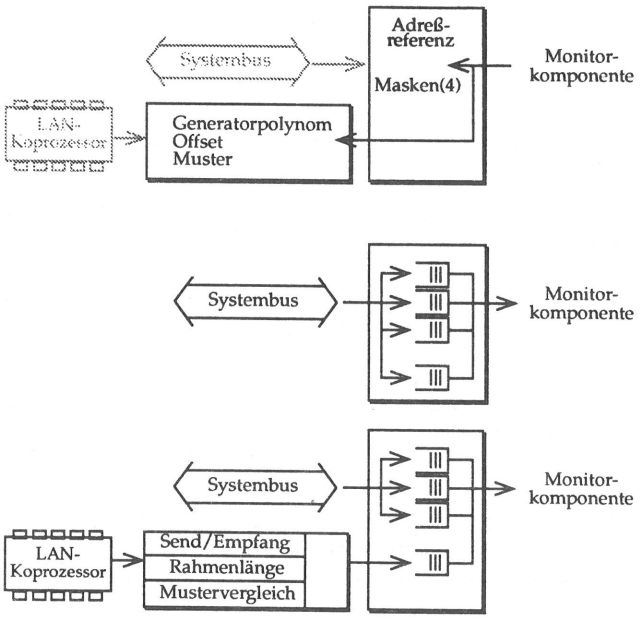


Bild 5.12: Meßfühler und Anschluß an die Monitorkomponente: Initialisierung (oben), Messung nur am Systembus (Mitte) und Messung an Systembus und LAN-Koprozessor (unten)

5.2 Meßstatements

Das Monitor-Register ist über einen festen Adreßbereich im System erreichbar, der Zugriff erfolgt immer wortweise. Bei den Betriebssystemen AT/RTX86 und iRMX86 kann die Initialisierung eines Zeigers auf das Monitorregister direkt erfolgen.

In den mit virtueller Speicherverwaltung arbeitenden Betriebssystemen RTX286, iRMX II und XENIX ist die Initialisierung eines Zeigers nicht möglich. Dort erfolgt die logisch-physikalische Adreßumsetzung in prozessorinternen Registern, die nur durch das Betriebssystem selbst veränderbar sind.

Bei RTX286 und iRMX II sind geeignete Systemaufrufe vorhanden, so daß lediglich bei Programmstart die Initialisierung über diese Aufrufe erfolgen muß. Dabei ist ein passenden Eintrag (engl. Descriptor) für die gewünschte Adresse mit erforderlicher Segmentlänge und Zugriffsrechten in der globalen Descriptortabelle (GDT) zu erzeugen. Wird der Zeiger auf das Monitorregister nicht mehr benötigt, so muß dieser unbedingt wieder aus der GDT gelöscht werden, da das Betriebssystem selbst diesen Descriptor nicht austragen kann und im gesamten System nur wenige unbelegte Descriptorplätze vorhanden sind.

Unter XENIX kann allein der Betriebssystemkern Descriptoren reservieren. Deshalb muß bewußt eine Sicherheitslücke in das Betriebssystem XENIX eingebaut werden, damit ein Anwendungsprogramm direkten Zugriff auf das Monitorregister erhält. Dabei wurde der bestehende IHI-Treiber um diese Funktionalität erweitert. Vorteilhaft hierbei ist, daß die erhaltene logische Adresse mit Systemzugriffsrechten markiert ist, so daß mit dieser Adresse jeder Prozeß auf die Hardware-Adresse schreiben darf. Möchten mehrere Prozesse auf das Monitorregister zugreifen, so kann diese logische Adresse an alle Prozesse publiziert werden.

Nach der Zeigerinitialisierung, die in der XENIX-Umgebung erst nach der erfolgreichen Öffnung der IHI-Schnittstelle erfolgen kann, sind über den Zeiger *ZeigerAufMonitorRegister* beliebige Wertzuweisungen möglich.

5.3 Meßkomponente

5.3.1 Steuerungsbaugruppe

Auf dieser Baugruppe läuft das meldungsbasierte Betriebssystem QOS (engl. Queue Operating System). QOS simuliert einen Mehrprogrammbetrieb durch die serielle Abarbeitung von Programmphasen, die vom zentralen Meldungsverteiler aufgrund der in einer Warteschlange gespeicherten Meldungen angesprungen werden. Sind alle Programme kooperativ, so wird die verfügbare Prozessorzeit bedarfsgesteuert an die Programme verteilt.

Neben Anwendungsprogrammen kennt QOS Treiberprogramme, die die Verwaltung der Schnittstellen (Konsolenanschluß über V.24, Lokales Netz und Hintergrundspeicher) übernehmen. Den größten Teil der Treiberprogramme stellt der Netztreiber mit den implementierten Schichten ISO 2a und ISO 2b dar, auf den eine Erweiterung für sicheren Datentransfer (vergl. Kap. 5.4.2) aufsetzt.

Zur Leistungsmessung existiert eine Applikation, die Befehle von der Konsole oder dem Lokalen Netz empfängt und in entsprechende Aktionen der Meßbaugruppe umsetzt.

5.3.2 Meßbaugruppe

Die in Mehrlagentechnik aufgebaute Baugruppe wird von außen über zwei Register angesprochen:

- Datenregister zum Schreiben der Initialisierungsdaten und Lesen der Meßdaten sowie
- Kommandoregister für die Befehle (nur beschreibbar) und dem Statusregister (nur lesbar).

Das Kommandoregister speichert den zuletzt geschriebenen Wert, wobei einzelne Binärstellen direkt zur Steuerung eines Automaten innerhalb der Baugruppe verwendet werden. Das Statusregister unterteilt sich in zwei Teile: die aktuelle Zustandsanzeige für die Meßbaugruppe und den Meßspeicher. Änderungen des internen Zustands dieser Baugruppe müssen periodisch abgefragt werden.

Beim Betrieb der Meßbaugruppe unterscheidet man 3 Phasen:

- Initialisierung — die Initialisierungsdaten werden von der Steuerung in das Datenregister geschrieben. Von dort aus werden sie über den Abfragebus an die Meßfühler transferiert. In dieser Betriebsphase liefert der Hardware-Automat lediglich die Meßfühleradressen.
- Meßphase — Der Automat übernimmt die Kontrolle über die Meßbaugruppe und frägt periodisch alle konfigurierten Meßfühler ab. Sind Daten im Meßfühler abrufbereit, so wird ein Informationswert übertragen und zusammen mit dem aktuellen Zeitstempel im Meßspeicher abgelegt.
- Transferphase — alle im Meßspeicher abgelegten Daten werden über den Rückwandbus ausgelesen und von der Meßapplikation unter QOS über die Schnittstelle übertragen.

Der zur Abfrage der Meßfühler zuständige Automat stellt auch bei der Initialisierung (*MESS* inaktiv in Bild 5.13) die Meßfühleradresse bereit. Somit wird immer ein Initialisierungswert (*D0..15*) an einen vom Automaten adressierten Meßfühler (*A0..2*) geschrieben. Sind in einen Meßfühler mehrere Werte zu laden, so kann dies nur durch aufeinanderfolgende Zyklen geschehen. Dies verlangt, daß Meßfühler mit nur einem Ladeplatz immer denselben Ladewert je Zyklus erhalten, während Meßfühler mit mehreren Ladewerten (z.B. dem Mustererkennungs-VLSI Baustein) die Ladewerte in definierter Reihenfolge erhalten.

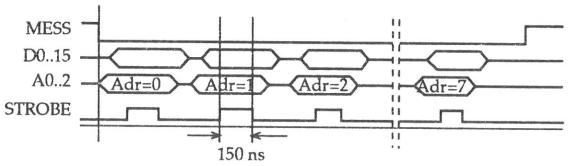


Bild 5.13: Initialisierungsphase (Zyklusgröße 8)

Nach erfolgter Initialisierungsphase schaltet die Steuerung befehlsgesteuert den Automaten in die Meßphase um. Es wird augenblicklich mit der Abfrage der Meßfühler begonnen. Diese Meßphase endet entweder durch Befehl von außen oder mit Erreichen des Speicherendes. Fehler, wie ein eventueller Zeitstempelüberlauf, werden in einem Statusregisterfeld angezeigt, behindern aber die laufende Messung nicht.

Der Meßspeicher ist FIFO-mäßig organisiert und speichert alle vom Abfragebus empfangenen 18 bit breiten Informationen zusammen mit dem 32 bit Zeitstempel und der 3 bit Adresse des Meßfühlers. Jeder Eintrag ist auf der Baugruppe 56 bit breit. Über einen FIFO-Erweiterungsbus kann der Meßspeicher von 2^{15} Einträgen in Schritten von jeweils 2^{16} Einträgen vergrößert werden. Der Füllstand der einzelnen FIFO-Blöcke wird im Statusregister angezeigt, so daß die Steuerung diesen an die zentrale Auswertestation melden kann.

Die 56 bit breiten Dateneinträge können in der Transferphase nicht auf einmal über den Rückwandbus übertragen werden. Deshalb wird jeder Dateneintrag durch 4 aufeinanderfolgende Worte von der Steuerung unter Mithilfe des Hardware-Automaten aus dem FIFO-Meßspeicher ausgelesen. Dabei adressiert der Automat einen Eintrag, und über spezielle Befehle kann dann ein Teil des FIFO-Eintrags gelesen werden. Mit dem letzten gelesenen Teil muß der Automat die Adresse im

FIFO-Speicher inkrementieren. Dieser Vorgang wird solange wiederholt, bis entweder das FIFO-Ende erreicht ist (erkenntlich im Statusregister) oder eine reservierte Endekennung im Informationsfeld gelesen wurde. Die zweite Methode hat den Vorteil, daß nicht immer der gesamte FIFO-Inhalt ausgelesen werden muß.

Der Zeitstempel ist als 32 bit Synchronzähler implementiert. Mit jedem Ende eines Abfragezyklus wird der Zähler inkrementiert. Durch den während der Meßphase gewählten Ablauf im Automaten hängt die Abfragezykluslänge, und damit auch der Zählerzyklus T, nur von der Anzahl der abgefragten Meßfühler ab:

$$T_f^n = \frac{2^{32}}{F} * 4 * n * f \qquad \text{(Gl.5)}$$

mit: F Oszillatorfrequenz (12,5 MHz)
n Anzahl der Meßfühler (2, 4, 8)
f Teilerfaktor (2, 8, 32, 64)

Nach dieser Zeit erfolgt ein neuer Zählerzyklus. Die sich ergebenden Zyklusdauern sind in Tabelle 14 gezeigt. Ein Zeitstempelüberlauf wird im Statusregister vermerkt.

Meßfühler	Teilerfaktor	Abfragezyklus	Zählerzyklus T (hh:mm:ss,ss)
2	2	640 ns	45:48,779
	64	20,48 µs	24:26:00,930
4	2	1,28 µs	1:31:37,558
	64	40,96 µs	48:52:01,860
8	2	2,56 µs	3:03:15,116
	64	81,92 µs	97:46:55,519

Tabelle 14: Zählerzyklus und zeitliche Auflösung

5.4 Realisierung der Meßsteuerung

5.4.1 Master-Slave-Betrieb

Im bestehenden verteilten Meßsystem wird jede dezentrale Aktion durch einen Befehl von der zentralen Station aus angestoßen. Der Befehl wird anschließend sofort quittiert. In der zentralen Station wird durch die ausgegebenen Befehle ein Zustandsmodell geführt, das über die Quittungsmeldungen abgesichert wird.

5.4.2 Kommunikationsprotokoll

Zur Prototypimplementierung stand nur ein Transportsystem mit Dienstzugang zu ISO TP4 und ISO 2b zur Verfügung. Die Implementierung einer Transportinstanz in der Steuerungsbaugruppe schied aus Aufwandsgründen und der fehlenden Gruppenadressierung aus, weshalb die in Bild 4.9a vorgestellte Schichtung mit LLC und einer Erweiterung gewählt wurde. Aufgrund verschie-

dener Beschränkungen der beteiligten Systeme entstand ein fehlertolerantes, auf LLC Class I basierendes verbindungsähnliches Protokoll mit Quittierungsmechanismus auch bei Gruppenadressierung [153].

Bild 5.14 zeigt den Rahmenaufbau. Jedes zu übertragende Paket wird mit einem ISO 2b Datagramm verschickt. Ein zusätzlicher, nicht normkonformer, Steuerkopf (engl. Header) enthält die Zuordnungsinformation zu den adressierten Applikationsprogrammen (engl. Destination Application_Access Point, DA_AP und Source Application_Access Point, SA_AP) in den *DA_AP*, *SA_AP*-Feldern, eine Pakettypkennung *MSGTYP* sowie die Reihenfolgenummer in Feld *SEQ*.

Als Dienstzugangspunkte (engl. Service Access Points, SAP) sind zwei SAPs nötig, da die Auswertestation einen Sende- und einen Empfangsprozeß mit je einem zugeordneten SAP benützt: über einen Schicht 2b Dienstzugangspunkt werden die Befehle gesendet und über den anderen Dienstzugangspunkt die Quittungen empfangen. Zum Aufbau einer Kommunikationsbeziehung zwischen Auswertestation und Monitorkomponenten ist der erfolgreiche Austausch eines ersten Datenpaketes erforderlich. Damit wird ein Eintrag in Tabellen der Erweiterungsschicht-Instanzen erzeugt, die alle erforderlichen Informationen zur Zuordnung dieser Beziehung an die lokalen Prozesse sowie die aktuellen Reihenfolgenummern umfassen. Nun können bidirektional Pakete ausgetauscht werden; jedes Paket wird mit einem Quittungspaket beantwortet. Die Einträge bleiben solange in den Tabellen bestehen, bis ein Paket vom *MSGTYP* CLOSE erfolgreich übertragen wurde.

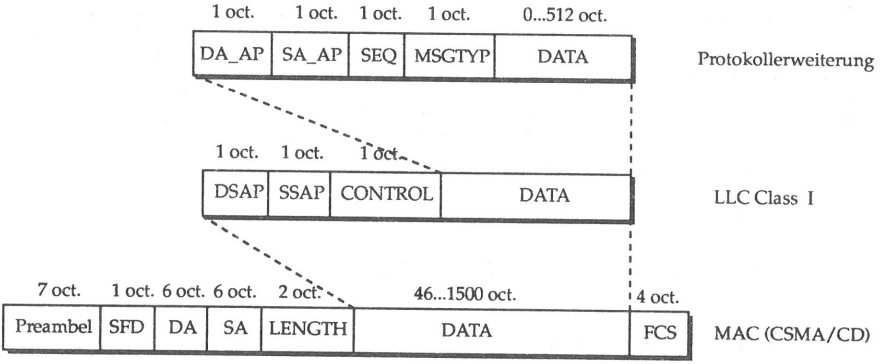


Bild 5.14: Aufbau des LLC I-Rahmens und eines Protokollerweiterungsrahmens

Im vorliegenden Netz auf Basis von CSMA/CD können Pakete kollidieren, gestört werden oder beim Empfang verloren gehen. Deshalb wird jedes gesendete Paket durch eine Auszeit (engl. Time Out) überwacht. Wurde innerhalb dieser Zeit kein Quittungspaket empfangen, so kann dieses Paket mehrmals wiederholt werden, bevor die Übertragung mit einem Fehler abbricht. Durch Konfiguration der Wiederholzeiten und der Versuchsanzahl läßt sich die Erweiterungsschicht-Instanz an unterschiedliche Konfigurationen anpassen.

Die Behandlung der Pakete bei Fehlern ist in Bild 5.15 skizziert. Jedes erfolgreich übertragene Paket wird an das Applikationsprogramm weitergereicht und quittiert. Geht die Quittung verloren, so bedeutet dies, daß der Sender das Paket wiederholt. Dieses wiederholte Paket wird nicht noch-

mals an die Applikation gereicht, sondern nur quittiert. Zur Entscheidung, ob ein Paket schon einmal weitergereicht wurde, dient das Feld *SEQ*, das die laufende Reihenfolgenummer enthält und einen starren Synchronisationsmechanismus (Hand Shake) bei Fenstergröße eins erzwingt.

Im Gegensatz zu LLC Class III (Anhang zu [98]), bei dem die Verbindung nicht mit einem CLOSE-Paket sondern zeitgesteuert abgebaut wird, beinhaltet diese Erweiterung auch den sicheren Transfer von Paketen mit Gruppenadressen. Dabei wird dasselbe Verfahren wie im Falle eindeutiger Adressierung angewandt. Ergänzend kommt hinzu, daß nun nicht nur eine Quittung zurückkommt, sondern von jedem Gruppenmitglied eine. Die Protokollerweiterung erlaubt in diesem Fall, daß der Sender eines Multicast-Rahmens vorgibt, wieviele Quittungen er erwartet. Werden alle Quittungen innerhalb eines vorgebbaren Zeitraumes empfangen, so wird dieser Transfer erfolgreich beendet. Fehlen einzelne Quittungen, so bricht der Versuch wieder nach einer vorgebbaren Wiederholungsanzahl ab. Zusätzlich können die Adressen der sich meldenden Zielstationen

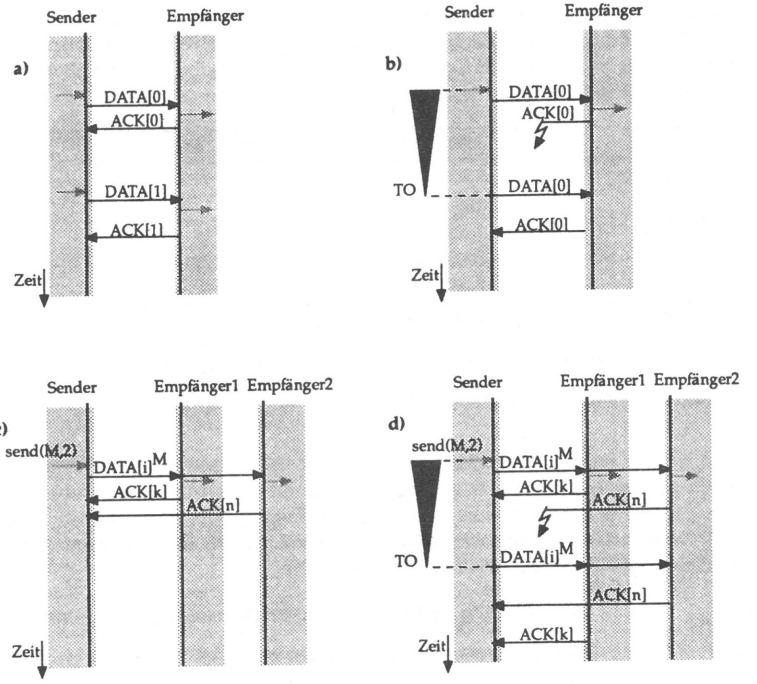


Bild 5.15: Szenarien beim Pakettransfer mit LLC Class I und Erweiterungsprotokoll:
a) fehlerfreier und b) gestörter Transfer mit Wiederholung,
c) fehlerfreier und d) gestörter Multicast-Transfer mit Wiederholung an 2 Empfänger
(M Multicast, [i] i.tes Paket in Reihenfolge)

abgefragt werden, um die nicht quittierenden Stationen zu identifizieren. Diese Stationsidentifizierung ist möglich, weil die antwortenden Stationen den Multicast-Rahmen mit einem Unicast-Rahmen beantworten.

Das von [33] beschriebene Pufferproblem bei Multicast-Empfang kann hier auftreten, da alle angesprochenen Stationen gemäß Bild 5.15c und 5.15d sofort mit einem Quittungspaket antworten. Treffen zu viele Pakete beim Sender ein, so kommt es zwangsläufig zu einem Paketverlust durch Pufferengpässe. In dieser Prototypimplementierung wird kein Mechanismus (vergl. [209, 111]) angewandt, um dieses Problem zu lösen. Vielmehr wird versucht, die Anzahl von Teilnehmern in der Gruppe möglichst klein zu halten, um mit den verfügbaren Empfangspuffern alle eintreffenden Quittungen zu speichern.

5.4.3 Befehlssatz

Der in der Monitorkomponente eingesetzte Prozessor verfügt über eigene Programme im Festwertspeicher. Mit dem Einschalten wird die Monitorkomponente initialisiert und in einen passiven Zustand versetzt, so daß die Befehle von der zentralen Station nur Aktionsauslöser für die Meßbaugruppe darstellen. Die Befehle sind entsprechend den Betriebsphasen unterteilt (vergl. Tabelle 15).

Phase	Befehl	Kodierung (Befehl, Meldung)
Initialisierung	ABORT	a, A
	INIT	i, I+J
Messung	PROGFREQ	p, P
	MEASURE	m, M
	STATUS	q, Q
Transfer	TRANSMIT	t, T
	RETRANSMIT	r, R

Tabelle 15: Befehle und Kodierungen

Jeder Befehl wird im Datenteil eines Paketes von der zentralen Auswertestation an die Monitorkomponenten übertragen. Sofern nötig, folgen auf die Kennbuchstaben die Parameter in der Form, wie sie in die Meßbaugruppe zu laden sind. Ist die Länge der Parameterliste variabel, so gibt der erste Wert nach dem Kennbuchstaben die aktuelle Länge an. Die Befehls- und Meldungssyntax in Anlehnung an die Backus-Naur Form lautet wie folgt:

Befehl:	Befehlskennung | Befehlskennung Wert | Befehlskennung Länge Wertfolge;
Meldung:	Meldungskennung | Meldungskennung Wert | Meldungskennung Länge Wertfolge;
Wertfolge:	Wert | Wertfolge Wert ;
Länge:	Wert ;
Befehlskennung:	'a' | 'i' | 'm' | 'p' | 'q' | 'r' | 't' ;
Meldungskennung:	'A' | 'E' | 'I' | 'J' | 'M' | 'P' | 'Q' | 'R' | 'T' | 'Y' ;
Wert:	0..65535 ;

Diese Syntax erlaubt für jede Befehlskennung Parameter. Werden bei der Befehlsabarbeitung in der Monitorkomponente Fehler erkannt, so führt dies zur Anzeige durch Fehlermeldungen ('Y': falscher Befehl und 'E': Befehl korrupt). Fehlerhaft von der Monitorkomponente zur Auswertestation übertragene Datenpakete können während der Transferphase durch den RETRANSMIT-Befehl erneut angefordert werden.

Ein Sonderfall besteht beim Initialisieren der Meßbaugruppe. Der Meßspeicher kann auf unterschiedliche Speichergrößen aufgerüstet werden. Dies verlangt, daß der Initialisierungsbefehl (i) quittiert (I) und nach jedem initialisierten Speicherblock eine Zwischenmeldung (J) mit der bisherigen FIFO-Länge transferiert und damit der zentralen Auswertestation die aktuelle Gesamtlänge bekannt gemacht wird.

5.5 Zusammenfassung der Ereignisspuren

Bevor die Zusammenfassung der einzelnen Ereignisspuren erfolgt, muß die Einhaltung der Monotonie der Zeitstempel geprüft werden und eine eventuelle Korrektur nach Überlauf im Zeitstempelfeld stattfinden. Selbst bei langen Messungen ist es sehr unwahrscheinlich, daß mehr als ein Zählerzyklus ohne Meßwert abgelaufen ist. Tabelle 14 zufolge würde dies eine Erzeugungspause von mindestens 45 min 48,779 s bedeuten. In diesem Fall muß auf dem Auswerterechner der vorzeichenlose 32 bit Wert des Zählers (entspricht dem größten Ganzzahl-Datentyp in der Kodierungssprache C) in einen Gleitkommawert konvertiert werden.

5.5.1 Korrelation

Zur Berechnung der Korrelation werden nur die Ereignisse verwendet, die als SYNC-Ereignisse in der Ereignisspur gekennzeichnet sind, d.h. die der VLSI-Baustein beim Mustererkennungsprozeß am Netzzugang erzeugt hatte.

Die Spurenzuordnung erfolgt mit einem Programm, das in folgenden Schritten vorgeht:

- Extrahieren der SYNC-Abstände in allen Spuren,
- Festlegen der statischen und dynamischen Vergleichsintervalle,
- Korrelation mit den aktuellen Vergleichssintervallen,
- Eventuell erneute Korrelation an anderer Stelle innerhalb der Spuren und
- Erzeugung der Referenzspur.

Der gesamte Rechenschritt wird über Vorgaben gesteuert. Dazu wird eine Steuerdatei eingelesen, die mit Schlüsselworten die aktuellen Vorgaben beschreibt:

TRACES	Anzahl zu bearbeitender Ereignisspuren
TIME_RESOLUTION	zeitliche Auflösung jeder einzelnen lokalen Uhr
MAX_MF	maximale Anzahl abgefragter Meßfühler je dezentraler Monitorkomponente

BUS_POSITION örtlicher Abstand der einzelnen Stationen zum Referenzpunkt

Außerdem sind sämtliche Parameter wiederum durch Schlüsselworte angebbar:

REFERENZ [datei | mittelwert | varianzminimierung]
Referenzspur wird generiert aus Datei, Mittelwert aller SYNCs oder Mittelwert mit sukzessiver Verringerung der Varianz

KORRELATION [spur-spur | muster-spur | alles]
Korrelation der Spuren gegeneinander, Spuren gegen Referenzspur oder beides

MAX_DIFFERENZ [prozentzahl]
erzeugt intern ein zusätzliches (fehlendes) SYNC, falls:
SYNC_Abstand > (1+*prozentzahl* / 100)**mittel*
und *mittel* der aktuelle mittlere SYNC_Abstand ist

FEHLER [zahl] bei *zahl* fehlenden SYNCs bricht Programm ab

WIEDERHOLUNG [zahl]
überlese *zahl* SYNCs, bevor erneut Korrelation versucht wird

AUTOMATIK [aus | ein]
Vergleichsintervallängen vorgegeben oder intern gewählt

S_xxx Größen für statisches Intervall *S*

D_xxx Größen für dynamisches Intervall *D*

Ausgehend von den SYNC-Ereignissen wird eine diskrete Folge aus den einzelnen Abständen zwischen den SYNC-Ereignissen jeder Spur gebildet. Jeweils zwei Abstandsfolgen (d.h. Ereignisspuren) werden dann miteinander korreliert. Mit dem statischen Intervall *S* wird ein Bereich von SYNC-Ereignissen vorgegeben, innerhalb dessen eine Korrelationsrechnung durchgeführt wird. In diesem Fenster wird ein kleineres Intervall (das dynamische Intervall *D*) verwendet, um als Stützstellen der eigentlichen Korrelationsrechnung zu dienen.

Die Vorgehensweise bei der Suche nach dem Maximum des Korrelationskoeffizienten ist, zuerst mit einem kleinen dynamischen Intervall zu beginnen. Durch schrittweises Vergrößern dieses Intervalls wird eine längere Folge von Stützstellen betrachtet. Der Rechenvorgang kann abgebrochen werden, sobald die Entscheidung für einen Verschiebewert eindeutig wird. Als Vorgaben dienen:

- Startpunkt und Länge des statischen Intervalls und
- Startpunkt und Länge des dynamischen Intervalls.

Zu jeder obigen Vorgabe ist ein Tripel aus (Minimalwert, Maximalwert, Inkrement) anzugeben.

Das Vergrößern des dynamischem Intervalls innerhalb eines statischen Intervalls und Wiederholung des gesamten Vorgangs bei verschobenem dynamischen Intervall (Bild 5.16) bedeutet, daß viele Korrelationskoeffizienten berechnet werden. Der Benutzer kann über die Steuerdatei alle Parameter zur Korrelationsberechnung vorgeben. Dies verlangt vom Benutzer oft zu viel Detailwissen, weshalb eine AUTOMATIK eingebaut wurde.

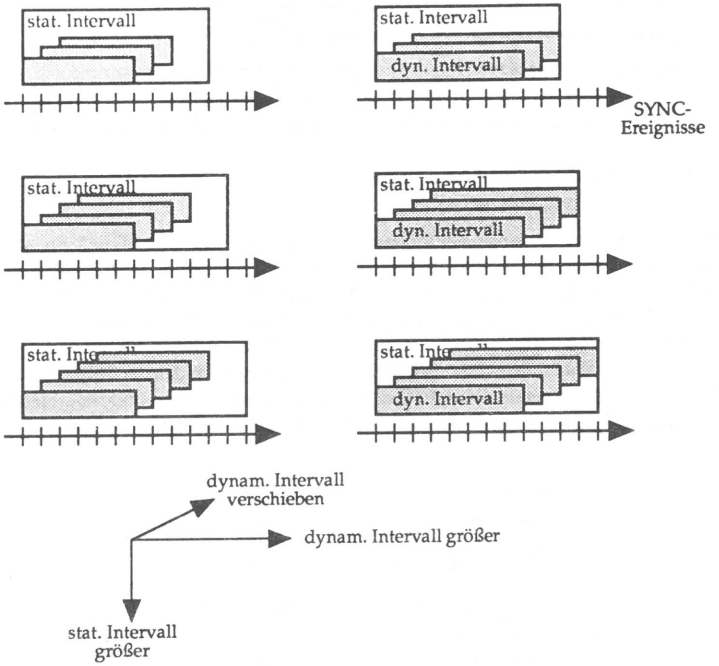


Bild 5.16: Statisches und dynamisches Intervall bei der Berechnung des Korrelationskoeffizienten je Abstandsfolge

Der Verschiebewert mit der größten Wahrscheinlichkeit (dem größten Korrelationskoeffizienten, vergl. Bild 5.17) wird in eine Matrix eingetragen (Bild 5.18). Treten Widersprüche in der Matrix auf, so wird die Berechnung beendet, sofern nicht WIEDERHOLUNG der Korrelationsrechnung an anderer Stelle der Ereignisspuren verlangt wurde.

```
korreliere SPUR.001 mit SPUR.003
Gesamt-Ergebnis:
  -2    6  50.0% **************************************************
  -1    3  25.0% ************************
   1    1   8.3% *******
   3    2  16.7% ***************
-->Wahl:  -2
```

Bild 5.17: Beispielergebnis der Korrelation zweier Ereignisspuren

	Spur 1	Spur 2	Spur 3
Spur 1	0	1	2
Spur 2	-1	0	1
Spur 3	-2	-1	0

Bild 5.18: Verschiebematrix bei drei Spuren

Nachdem die Zuordnung der einzelnen Spuren gefunden wurde, wird aus dem Mittelwert der Zeitstempelwerte der zugeordneten SYNC-Ereignisse ein SYNC-Eintrag in der Referenzspur errechnet (vergl. Bild 4.12). Die Erfassungszeit eines SYNC-Ereignisses *n* wird mit T_n bezeichnet. Das zu diesem Ereigniszeitpunkt zugehörige Genauigkeitsintervall $[T_n\text{-}d, T_n\text{+}d]$ beschreibt den Zeitbereich, innerhalb dessen ein Zeitstempelwert gleichverteilt liegen könnte. Einfluß auf die Zeitabweichung *d* haben die freilaufenden Uhren (ΔZ), die Verzögerungen in den Meßfühlern und die *s* wartenden Einträge am Abfragebus. Weil die Zeiten T_n^i mit unabhängigen und freilaufenden Zählern gemessen werden, läßt sich kein Vertrauensintervall für die Referenzzeit angeben.

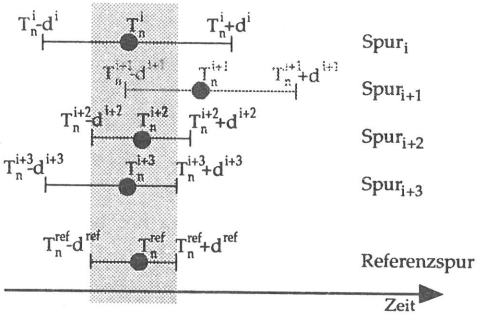


Bild 5.19: Mittelwertbildung für Referenzspur mit Varianzminimierung (Eintrag aus Spur i+1 wird aufgrund seiner starken Abweichung nicht zur Mittelwertbildung herangezogen)

Alle SYNC-Ereignisse, die nicht zu stark von den Werten der anderen SYNCs entfernt liegen, werden zur Mittelwertbildung herangezogen. Durch diese Varianzminimierung läßt sich der Einfluß stark abweichender Uhren bei der Referenzspurbildung vermeiden. Allerdings bedeutet dies auch, daß damit die Mehrzahl der Uhren, die einheitliche Zeitwerte liefern, die Referenzzeit bestimmen, selbst wenn diese Uhren mit falscher Frequenz oder starken Frequenzschwankungen laufen. Damit ergibt sich insgesamt jedoch eine verbesserte Gesamtgenauigkeit (vergl. Bild 5.19 und [39]) unter der Annahme, daß die absolute Frequenz der Uhrenoszillatoren genau und die Frequenzabweichungen klein sind.

5.5.2 Interpolation

Dieser Rechenschritt wird von demselben Programm durchgeführt, welches auch die Korrelation bestimmt. Die für diesen Schritt gültigen Eingabeparameter sind:

INTERPOLATION [linear | akima | spline]
Art der Zeitinterpolation zwischen zwei SYNCs

Im einfachsten Fall wird der Zeitverlauf zwischen zwei SYNCs linear angenähert. Die AKIMA-Interpolation verwendet ein Polynom dritten Grades zwischen den Stützstellen, verzichtet aber (im Gegensatz zur Spline-Interpolation) auf die Stetigkeit der 2.Ableitung in den Stützstellen und betrachtet neben dem Polygonpunkt noch jeweils zwei weitere Punkte zu beiden Seiten. Als Interpolationspolynom verwendet man:

$$P_i(x_i) = A_i + B_i(x - x_i) + C_i(x - x_i)^2 + D_i(x - x_i)^3 \qquad \text{(Gl.6)}$$

mit den Koeffizienten:

$$A_i = y_i$$

$$B_i = t_i$$

$$C_i = \frac{3m_i - 2t_i - t_{i+1}}{x_{i+1} - x_i}$$

$$D_i = \frac{t_i + t_{i+1} - 2m_i}{(x_{i+1} - x_i)^2}$$

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

$$t_i = \frac{|m_{i+1} - m_i| * m_{i-1} + |m_{i-2} - m_{i-1}| * m_i}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|} \qquad \text{(Gl.7)}$$

Bei i=0 und i=n werden die fehlenden Stützstellen am Rand i= (-2, -1, n+1, n+2) interpoliert. Für jede Ereignisspur werden bei der AKIMA-Interpolation für jedes SYNC-Intervall die gültigen Interpolationsparameter bestimmt.

Während einer Messung sollten keine Ereignisse verloren gehen. Trotzdem ist der Algorithmus in der Lage, fehlende SYNC-Ereignisse (vergl. Bild 5.20a) zu erkennen und selbständig in die jeweilige Ereignisspur einzufügen. Als Auftrittszeitpunkt des erzeugten SYNC-Ereignis wird momentan der errechnete Referenzzeitpunkt eingesetzt. Zur Erkennung fehlender SYNCs wird ein Intervall (MAX_DIFFERENZ) angegeben, innerhalb dessen weitere SYNCs folgen müssen. Ist dies nicht der Fall, so wird ein zusätzliches SYNC eingefügt. Treten zu viele SYNC-Verluste in Folge auf (Parameter FEHLER), so wird vorzeitig terminiert.

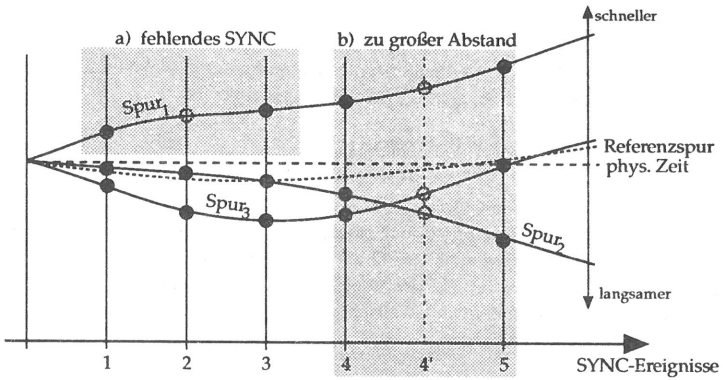


Bild 5.20: Sonderfälle bei Berechnung der Referenzspur: a) fehlendes SYNC in einer Spur, b) 'fehlende' SYNCs in allen Spuren aufgrund zu großer SYNC-Abstände

Der Fall b) aus Bild 5.20 kann erkannt werden, indem wiederum der maximale Abstand zwischen den SYNC-Ereignissen überprüft wird. Da nicht feststeht, ob diese Lücke durch einen Fehler oder gewollt entstanden ist, wird in der vorliegenden Implementierung dieser Fall nicht gesondert behandelt. Die Erzeugung von zusätzlichen SYNCs in allen Spuren ist mit Problemen behaftet ist: eine Möglichkeit wäre, diese Lücke durch Anpassung eines Kurvensegments an die beiden Randstellen zu schließen und an dazwischenliegenden Punkten virtuelle SYNCs (SYNC-Ereignis 4' in Bild 5.20) zu erzeugen.

5.5.3 Sortierung

Für jedes SYNC-Intervall werden die normalen Ereigniseinträge aus allen Spuren entsprechend ihrer interpolierten Auftrittszeit zu einer Ausgangsspur zusammengefaßt. Dieser Vorgang wiederholt sich für alle SYNC-Intervalle, bis keine Ereigniseinträge mehr vorhanden sind. Die Ausgabe bildet die globale Ereignisspur.

Durch eine Transformation der lokalen Meßfühleradresse (l,s) in eine globale Adresse g bleibt für die anschließende Auswertung der Generierungsort erhalten:

lokaler Meßfühler l aus Ereignisspur s $\rightarrow$ globaler Meßfühler g

mit: $g = (s * MAX\_MF) + l$ $\qquad g \in (0...2^8-1)$ (Gl.8)

MAX_MF gibt die maximal erlaubte Anzahl angeschlossener und abgefragter Meßfühler je Monitorkomponente an. Zusätzlich ist der Wertebereich von g nach oben durch die momentan gewählte rechnerinterne Darstellung eines Ereigniseintrages begrenzt und liegt bei $g = 2^8-1$, wodurch mit der Prototypimplementierung höchstens 64 (MAX_MF=4) bzw. 32 (MAX_MF=8) Monitorkomponenten betreibbar wären.

5.6 Implementierung des Statistikinterpreters

Zur statistischen Auswertung wird ein zweistufiger Ansatz verwendet:

- Parser zur Ereignisidentifikation mit Aufbau der Referenzdatenbank,
- statistische Auswertung (Interpreter) basierend auf dieser Referenzdatenbank.

5.6.1 Aufbau der Referenzdatenbank aus den Meßdaten

Die zur Ereignisbeschreibung verwendete kontextfreie Datenbeschreibungssprache (engl. Data Description Language, DDL) vom Chomsky-Typ 2 entsprechend Kap. 4.4.1 kann unter Verwendung linksrekursiver Ableitungsregeln (LALR(1)) durch einen endlichen Automaten abgearbeitet werden [120]. Für diesen Sprachtyp sind Hilfsmittel verfügbar, die beim Einlesen und Erkennen einzelner Tokens und Aufbau eines Kellerautomaten zur semantischen Prüfung helfen. Dazu werden die Syntaxregeln in einer maschinenlesbaren Form der BNF angegeben (siehe Anhang A). In dieser Sprachbeschreibung sind keine eigendefinierten Namen zulässig, so daß Werte immer in einer erlaubten Wertedarstellung (binär, dezimal oder hexadezimal) angegeben werden müssen. Dieser Nachteil läßt sich jedoch sehr einfach durch vorausgehendes Anwenden des C Präprozessors beseitigen.

Die Adressierung der Meßfühler erfolgt über das optionale Schlüsselwort FIFO in der DDL-Beschreibung. Falls angegeben, muß es als erste Feldbeschreibung (eingeleitet durch das Token 'fifo') stehen.

Während des Einlesevorgangs der DDL-Beschreibung werden vom Parser intern zwei Masken erstellt, die zur Identifikation eines Ereigniseintrags gebraucht werden. Die Zuordnung eines Ereignisses zu einem Feinmuster ist korrekt, wenn folgende Gleichung erfüllt ist:

$$(\text{Informationsfeld } \mathbf{UND} \text{ FeldMaske}) \ \mathbf{EXOR} \ \text{WertMaske} \equiv 0 \qquad \text{(Gl.9)}$$

Die Masken ergeben sich aus der Feinmusterdefinition von *ProzedurStart* nach Bild 4.30:

```
ProzedurStart:    FeldMaske = 11 111111 00000000b = 0xff00
                  WertMaske = 00 110000 00000000b = 0x2000
```

Für jedes initialisierte Feld im *Informationsfeld* eines Feinmusters wird ein Bit in der *FeldMaske* gesetzt. Die diesem Initialisierungswert entsprechenden Binärdarstellungen werden positionsrichtig zu der *WertMaske* zusammengebaut. Die logische Operation entsprechend (Gl.9) wird dann auf das *Informationsfeld* jedes Spurereignisses angewandt. Ist das Ergebnis Null, so wird die relative Position dieses Feinmusters in der Referenzdatei abgelegt. Nachdem die globale Ereignisspur komplett abgearbeitet wurde, stehen alle gefundenen Positionen in zeitlich monoton steigender Auftrittsfolge in dieser Referenzdatei (Bild 5.21). Durch die Verwendung des Feinmusternamens als Dateiname ergibt sich eine betriebssystemabhängige Längenbeschränkung der Musternamen.

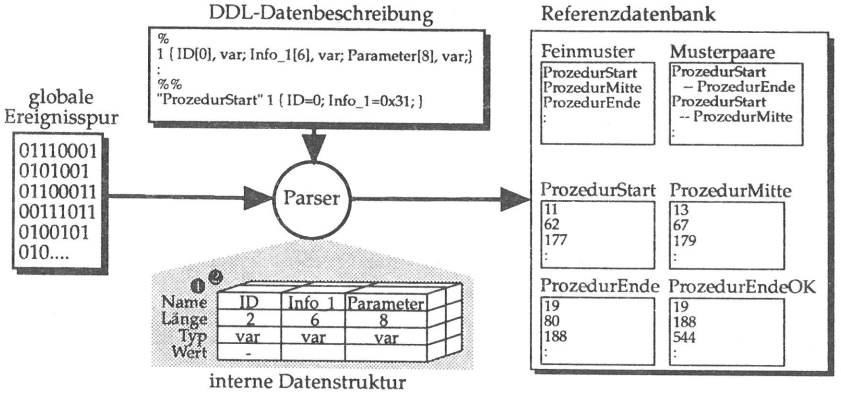


Bild 5.21: Datenfluß beim Aufbau der Referenzdatenbank

5.6.2 Statistische Interpretation der Daten aus der Referenzdatenbank

Nach dem Aufbau der Referenzdatenbank sind unterschiedliche Möglichkeiten der Auswertung vorhanden, die spezifisch sind für:

- Ereignistypen,
- Ereignispaare oder
- Ereignisserien.

Zur Umrechnung der gemessenen Zählerwerte in absolute Zeiten werden die Anzahl abgefragter Meßfühler und die lokalen Zykluszeiten benötigt

5.6.2.1 Statistische Auswertung bezüglich Ereignistypen

Bei der Typanalyse werden für jedes gewünschte Feinmuster die Auftrittsabstände zwischen den Auftrittszeitpunkten betrachtet. Neben der Häufigkeitsbestimmung beschränkt sich die Berechnung auf die grundlegenden statistischen Größen:

- Extremwerte (Minimalwert, Maximalwert),
- arithmetischer Mittelwert,
- Varianz,
- n%-Quantile (Abstand, unterhalb dessen n=30, 50, 90% aller Wert liegen),
- Modalwert (Wert und Anzahl des häufigsten Abstands) sowie
- Histogramme.

Bei Berechnung der Quantile kommt der von [108] beschriebene P^2-Algorithmus zur Anwendung. Dabei werden fünf Markierer beim Durchlauf durch die Daten mitgeführt, aus denen am Ende die Quantile bestimmbar sind. Die Genauigkeits- und Fehlerabschätzung dieses heuristischen Algorithmus findet sich in [108].

Derselbe Algorithmus könnte auch bei der Berechnung von Median und Histogrammen angewandt werden, stattdessen reduziert eine Näherungsrechnung die Speicheranforderung bei der Medianberechnung. Anstelle alle Abstandswerte in Listen zu speichern, wird jeder auftretende Abstandswert über eine empirisch gefundene Glättungsmaske auf eine Intervallgrenze abgerundet. Dadurch bilden sich Zählerklassen, in die alle Abstandswerte eingetragen werden. Gibt es noch keine geeignete Klasse, so wird eine neue Zählerklasse angelegt. Nach einmaligem Durchlaufen der Ereignisspur liegt eine Liste von Zählerwerten vor, deren Maximum die häufigste Abstandszeit, d.h. den Median, angibt. Zusätzlich zum Median läßt sich aus der sortierten Liste eine Klassenverteilung mit konstanter Klassenbreite aber unbestimmter Klassenanzahl bilden, die nach einmaligen Durchlauf entstanden ist.

Eine weitere Möglichkeit zur Histogrammbildung mit quasi-logarithmischer Skala besteht, wenn man das Intervall zwischen Minimal- und Maximalwert in 2^{16} Abschnitte aufteilt. Als obere Klassenschranke K_i dient folgender Wert:

$$K_i = Min + K_{Schritt} * 2^i \qquad (0 \le i \le 16) \qquad \text{(Gl.10)}$$

und

$$K_{Schritt} = \frac{Max - Min}{2^{16}} \qquad \text{(Gl.11)}$$

Gegenüber der ersten Möglichkeit liefert diese Berechnung immer 17 Klassen (Min sowie K_i als oberen Klassenwert), verlangt aber zwei Durchläufe, da Minimal- und Maximalwert zur Berechnung der Schrittweite $K_{Schritt}$ bekannt sein müssen. Vorteilig an der logarithmischen Skaleneinteilung ist, daß die Berechnung im Programm mit einfachen Schiebebefehlen (Zeiten als Ganzzahlwerte) schneller durchgeführt werden kann. Zudem zeigt das Histogramm eine feine zeitliche Auflösung im Bereich kleiner Zeiten, während die Auflösung mit größer werdenden Abständen schnell abnimmt. Daß diese Art der Skaleneinteilung nützlich ist, zeigt der Vergleich in Bild 5.22, wobei die logarithmische Darstellung mehr signifikante Klassen aufzeigt.

Im Fall der logarithmischen Einteilung sind die Abstandszeiten zwischen den Feinmusterpaaren PStart→PStop sowie bei PStop→PStart in mehrere Klassen eingeteilt, während bei linearer Einteilung eine Klasse fast alle Einträge enthält. Dabei spielt es keine Rolle, ob die Extremwerte dicht beieinander liegen (PStart→PStop) oder die Laufzeitunterschiede mehrere Größenordnungen betragen (PStop→PStart).

5.6.2.2 Statistische Auswertung bezüglich Ereignispaaren

Zur Abstandsmessung zwischen zwei Ereignissen müssen die jeweiligen Start- und Endemuster in der Ereignisspur einander zugeordnet werden. Da von einem Startmuster immer zum nächstfolgenden Endemuster gesucht wird, gibt es mehrere Zuordnungsprobleme, die zu erkennen sind:

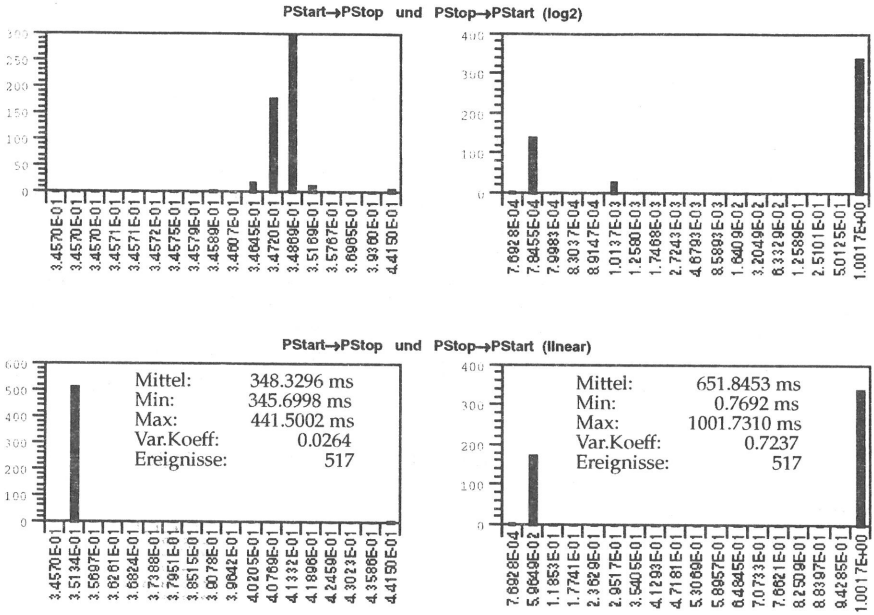


Bild 5.22: Logarithmische gegenüber linearer Klasseneinteilung
(links: PStart→PStop und rechts: PStop→PStart, nach Kap. 7.1.2)

- Mehrdeutigkeiten,
- Abhängigkeiten zwischen beiden Mustern und
- fehlende Start- bzw. Endemuster.

Mehrdeutigkeiten sind bei der Musterbeschreibung möglich, wenn ein Feinmuster (Muster1) eine Teilmenge eines anderen Feinmusters (Muster2) bildet (siehe Bild 5.23). Dieser Fall wird nicht ausgeschlossen. Bei der Paarzuordnung wird dies erkannt, da zum selben Zeitpunkt zwei unterschiedliche Muster vorliegen. Während der Auswertung wird das zeitgleiche Muster nicht zur Paarbildung herangezogen. Vielmehr wird das nächstliegende Endemuster gesucht.

Folgt auf das Startmuster (Muster1) ein weiteres Startmuster (siehe Markierung in Bild 5.23), so wird das im folgenden Startmuster enthaltene Muster2 als Endemuster erkannt. Ist eine solche Zuordnung nicht richtig, so ist eine geänderte Feinmusterdefinition zu wählen.

Wird ein Paar aus disjunkten Feinmustern gebildet, so sind drei Abhängigkeitsfälle denkbar:

- jedes Startmuster verlangt unmittelbar ein Endemuster (hart korreliert und synchron),
- jedes Startmuster bedingt unmittelbar mindestens ein Endemuster (korreliert und synchron),

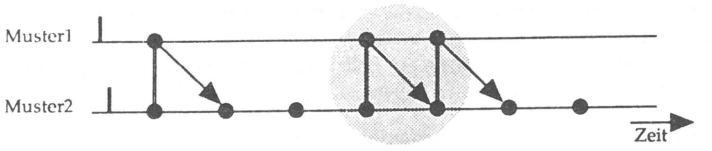


Bild 5.23: Paarbildung Muster1→Muster2 bei Mehrdeutigkeit der Musterdefinitionen (Muster1 ⊂ Muster2)

- zu jedem Startmuster gibt es ein Endemuster, die Auftrittszeit ist jedoch unbestimmt (Muster korreliert aber nicht synchron) oder
- Startmuster und Endemuster nicht paarweise (Muster nicht korreliert und nicht synchron).

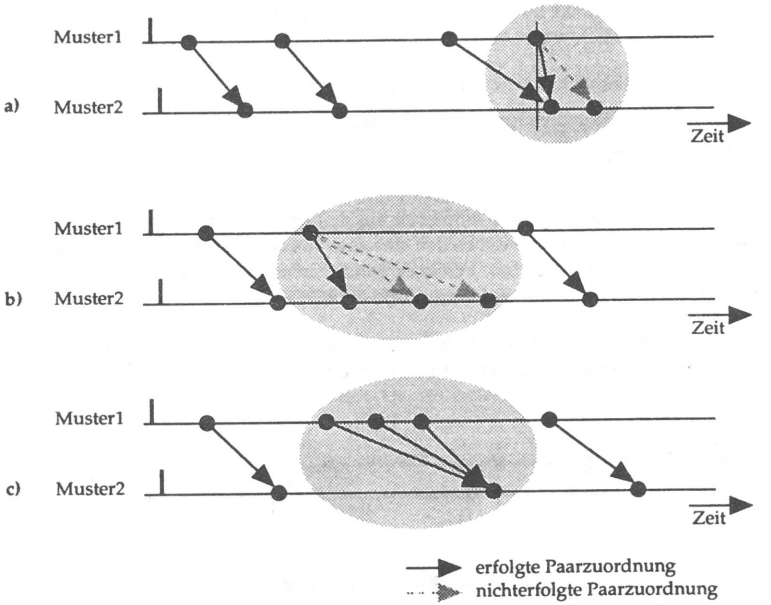


Bild 5.24: Eindeutigkeit bei disjunkten Mustern

Im trivialen Fall folgt immer auf ein Startmuster das Endemuster (Bild 5.24a, links). Folgen die Muster paarweise, jedoch nicht streng synchron, so wird bei Zuordnung zum nächsten Endeereignis eine falsche Paarbildung vorgenommen (Bild 5.24a, rechts).

Nicht eindeutige Zuordnungen gibt es immer dann, wenn die Muster nicht korreliert sind, d.h. nicht paarweise auftreten. Folgen auf das Startmuster mehrere Endemuster, so wird nur das erste Endemuster zur Paarbildung herangezogen. Damit werden alle weiteren Endemuster ignoriert (Bild 5.24b). Andererseits werden (gemäß Bild 5.24c) mehrere Paare von den Startmustern zu nur

einem Endemuster gebildet. Solche Fehlerfälle treten immer dann auf, wenn die Muster von asynchron laufenden Prozessen erzeugt werden. Da die Natur der Prozesse dem Interpreter nicht bekannt ist, werden auch keine Maßnahmen zur Behebung unternommen. Durch weitere Verfeinerung der Muster muß der Benutzer die Paarbildung in solchen Fällen unterstützen.

Liegt der Auftrittszeitpunkt des ersten Startmusters hinter dem ersten Endemuster, so wird bis zum nächsten Endemuster mit späterem Auftrittszeitpunkt gesucht. Fehlt zur Paarbildung das zugehörige Endemuster, so kann dieses unvollendete Paar nicht mitbewertet werden. Fehlende Muster zur Paarzuordnung sind immer dann zu erwarten, wenn die Messung bei bereits laufenden Objektsystemen gestartet wird oder vorzeitig endet.

Berechnet werden dieselben statistischen Größen wie bei der Typstatistik. Wird als Startmuster und Endemuster dasselbe Feinmuster verwendet, so entspricht dies der Typstatistik.

Eine Verallgemeinerung der Ereignispaare sind Serien von Ereignissen, die in fester Folge auftreten. Die bei der Paarbildung beschriebenen Probleme gelten auch hier, zusätzlich entstehen bei der Auswertung Sonderfälle, wenn

- die Ereignisses nicht direkt aufeinander folgen bzw.
- die Folge durch andere Ereignisse unterbrochen wird oder
- interne Ereignisse fehlen.

Wie bei Paaren wird verlangt, daß eine Serie aus den geforderten aufeinanderfolgenden Mustern besteht. Dies heißt nicht, daß zwischen den einzelnen Mustern keine weiteren Ereignisse in der Ereignisspur vorhanden sein dürfen. Fehlende Muster innerhalb einer Serie bedeuten, daß dieses Serienfragment nicht bewertet wird (vergl. Bild 5.25).

Die berechneten statistischen Größen für den zeitlichen Abstand ΔT vom ersten zum letzten Muster (Muster1..Muster4) sind wieder dieselben wie bei der Paar- und Typstatistik.

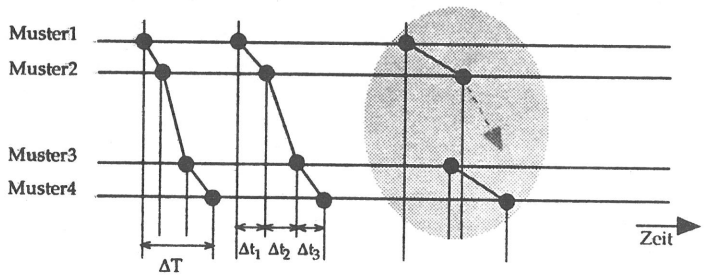


Bild 5.25: Zuordnung bei Musterserien

Zusätzlich läßt sich eine Musterserie als Verkettung von Einzelpaaren auffassen, deren Reihenfolge einzuhalten ist. Damit werden statistische Berechnungen über die internen Abstände Δt_i möglich.

5.6.2.3 Statistische Auswertung bezüglich gesamter Ereignisspur

Die Auswertung erfolgt durch das Interpreter-Programm, das als Eingabe die vom Parser erzeugte Referenzdatenbank neben der binären Datei der globalen Ereignisspur erwartet.

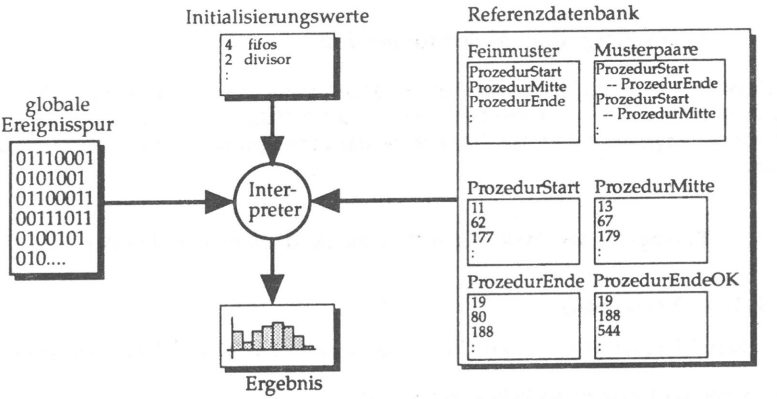


Bild 5.26: Datenfluß beim Interpretieren der Ereignisspur

Als Resultat wird eine Ausgabedatei erstellt, die parametergesteuert Typ-, Paar- und Serienstatistik umfaßt und zusätzlich Statistiken über alle Ereignisse, Meßfühler und den Berechnungsaufwand erstellt.

6 Güte und Leistungsbewertung des Meßsystems

6.1 Erzeugung der Meßinformation

Im folgenden wird angegeben, wann erkannte Meßinformationen von der externen Monitorkomponente erfaßt werden können. Die Beeinflussung durch Meßstatements in den Objektsystemen kann aus den Angaben in Kap. 6.1.2 und der Menge der eingebauten Anweisungen abgeschätzt werden.

6.1.1 Erzeugung der Meßinformation durch Hardware-Meßfühler

6.1.1.1 Netzzugang

Wie in Kap. 5.1.1 gezeigt, gibt es zwei Zeitpunkte, in denen Meßinformationen erzeugt werden:

- sofort bei Erkennen des Rahmenendes und
- mit Statuseintrag im SCB durch den LAN-Koprozessor.

Die Übergabe am Rahmenende erfolgt nicht unmittelbar bei Rahmenende, da der LAN-Koprozessor spätestens 6 Taktzyklen nach Ende des aktuellen Rahmens den Status des vorherigen Rahmens im SCB ablegt. Damit wird auch erst zu diesem Zeitpunkt die Sende-/Empfangserkennung für den vorherigen Rahmen abgeschlossen. Als frühest möglicher Zeitpunkt wird deshalb die steigende Flanke im 7. Taktintervall (T_{K7}) benützt, um einen Schreibzugriff auf den nachgeschalteten FIFO-Speicher zu initiieren.

$$T_{K7} = 7 * t_{Cl} \qquad \text{(Gl.12)}$$

Beim verwendeten CSMA/CD-Medienzugriffsprotokoll beträgt die Bitübertragungsrate 10 Mbit, d.h t_{Cl} = 100 ns.

Die Erkennung eines erfolgreichen Transfers ist erst mit dem Schreibzugriff in den SCB möglich, der auch noch nach Beginn des folgenden Rahmens erfolgen kann. Eine Möglichkeit des aufgebauten Meßfühlers [169, 50] ist, daß immer das Ergebnis (erfolgreich gesendet oder empfangen) bis zum Ende des folgenden Rahmens zwischengespeichert wird. Bei minimalem Rahmenabstand ergibt sich deshalb bei minimal langen Folgerahmen (72 Octetts Länge):

$$T_{Rahmen} = t_{Cl} * 72 * 8 \qquad \text{(Gl.13)}$$

$$T_{K6} = 6 * t_{Cl} \qquad \text{(Gl.14)}$$

$$T_{Abstand} = 96 * t_{Cl} \qquad \text{(Gl.15)}$$

Aus (Gl.13-15) summiert sich die Zeit bis zum Zugriff auf den SCB bei minimal langen Rahmen:

$$T_{SCB} = T_{Abstand} + T_{Rahmen} + T_{K6} \qquad \text{(Gl.16)}$$

Die Zeit, die bis zum sicheren Erkennen eines erfolgreichen Transfers durch Abwarten des folgenden Rahmens vergeht, muß bei allen Meßfühlern am Netzzugang in gleicher Weise akzeptiert werden. Da mit diesem Meßfühler der Meßpunkt immer um die Übertragungszeit eines beliebigen Rahmens verschoben ist, hängt der Erfassungszeitpunkt in der nachgeschalteten Monitorkomponente direkt vom Netzverkehr ab. Zusätzlich ist die Zeit zwischen zwei Rahmenenden bei CSMA/CD nicht für alle Stationen dieselbe. Je nach Busposition und Transferrichtung wird die Zeit unterschiedlich erkannt (Bild 6.1). Wird ein Ringnetzwerk (z.B. Token Ring) eingesetzt, so sind die Abstandszeiten identisch. Speichert man *alle* Rahmenendezeiten in einer eigenen Ereignisspur ab, so kann daraus auf das exakte Ende des jeweils vorausgegangenen Rahmenendes rückgerechnet und damit der Auftrittszeitpunkt in allen Ereignisspuren korrigiert werden.

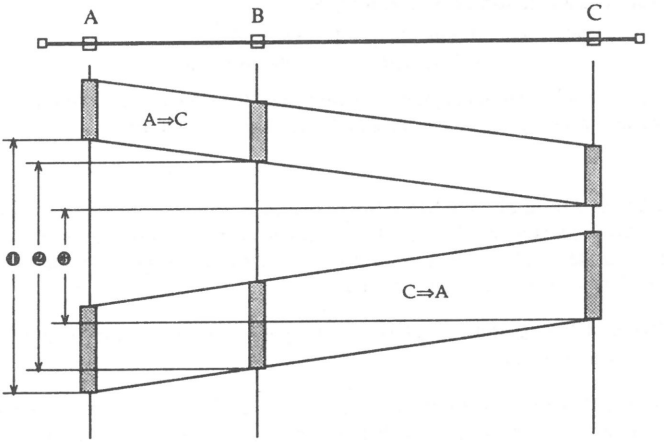


Bild 6.1: Einfluß der Busposition auf Meßpunktzuordnung bei CSMA/CD

Die Zuordnung zum Rahmen bleibt durch sofortiges Weitergeben der Information bei Schreibzugriff auf den SCB erhalten, jedoch kann in diesem Fall nur eine Abschätzung aufgrund der beobachteten Arbeitsweise des LAN-Koprozessors [92] abgegeben werden.

Annahme:

Empfangen: Interner FIFO-Puffer bei Rahmenende belegt mit Daten (schlechtester Fall)

- Zuerst wird mittels DMA-Betrieb der gesamte FIFO-Inhalt (16 Bytes) im Speicher abgelegt. Dabei muß die Zeit bis zum Erhalt der Buszugriffsberechtigung beachtet werden, die vor jeder DMA-Burstphase nötig ist. Nach 16/2 Speicherzyklen ist der FIFO-Puffer geleert, sofern keine weiteren Puffer aus der Empfangsliste angefordert werden müssen.
- Nach Rahmenende (CRS inaktiv) wird in den Rahmenblock des nächsten (noch leeren) Rahmens und den Verweisblock des aktuellen Empfangspuffers geschrieben. Dann wird

eine Statuskennung in den Verweisblock des aktuellen Puffers geschrieben. Zwei Lese- und einen Schreibzyklus später erfolgt nun der endgültige Zugriff auf den SCB.

Zeit = Buszuteilung + 8 Schreibzyklen (evtl. mehrere DMA-Bursts)
+ Buszuteilung + 3 Schreibzyklen
+ Buszuteilung + 1 Lesezyklus
+ Buszuteilung + 2 Schreibzyklen

- Zeitliche Lücken zwischen den einzelnen Zugriffen sind möglich. Wird der Bus nach der ersten Busanforderung nicht abgegeben, entfallen die folgenden Buszuteilungszeiten.

Senden: Kein Auftrag mehr in der Auftragsliste bei Sendeende

- LAN-Koprozessor schreibt in Auftragsblock.
- Danach das Statuswort im SCB eintragen.

Zeit = Buszuteilung + 1 Schreibzyklus
+ Buszuteilung + 1 Schreibzyklus

- Zeitliche Lücken zwischen diesen Schreibzugriffen sind möglich. Sofern die Buskontrolle zurückgegeben wird, sind die Buszuteilungszeiten zu beachten.

Der beschriebene Ablauf bis zum Schreiben in den SCB ist auf allen Objektsystemen bei Einsatz desselben LAN-Koprozessors ähnlich. Die einzelnen Verzögerungen infolge Buszuteilungs- und Speicherzykluszeiten sind abhängig von der jeweils verwendeten Hardware, der Koprozessorinitialisierung und der aktuellen Systembelastung.

Mit (Gl.12) und (Gl.16) sind die angebbaren Verzögerungen bekannt, die bis zum Einschreiben in den FIFO-Speicher meßbar sind.

Für den Einschreibevorgang wird für die Dauer $T_{Fifo} = 200$ ns die Meßinformation an den Eingängen angelegt, damit die Information sicher eingeschrieben werden kann. Gemäß [85] beträgt bei den verwendeten FIFO-Bauelementen bei vorher leerem Speicher die Durchfallzeit T_D:

$$T_D = t_{WLZ} + t_{WEF} + t_A \qquad \text{(Gl.17)}$$

mit: t_{WLZ} Aktivierung der Ausgänge (min. 20 ns, max. 40 ns)
t_{WEF} Schreibverzögerung (max. 60 ns)
t_A interne Zugriffszeit (max. 120 ns).

Für die unterschiedlichen Betriebsarten, d.h. Rahmenende (RE) bzw. Senden/Empfangen (SR), ergibt sich für die Verzögerung vom realen Meßpunkt bis zum Auslesezeitpunkt:

$$T_{RE} = T_{K7} + T_{Fifo} + T_D \qquad \text{(Gl.18)}$$

$$T_{SR} = T_{SCB} + T_{Fifo} + T_D \qquad \text{(Gl.19)}$$

Alle einzelnen Verzögerungswerte sind in Tabelle 16 zusammengefaßt.

Name	Zeit
T_{K6}	600 ns typ.
T_{K7}	700 ns typ.
T_{Fifo}	200 ns typ.
T_D	220 ns max.
T_{Rahmen}	≥ 57600 ns min.
$T_{Abstand}$	≥ 9600 ns min.
T_{SCB}	≤ 1230400 ns max.
T_{RE}	≤ 1120 ns
T_{SR}	≥ 67200 ns min.

Tabelle 16: Verzögerungen im Hardware-Meßfühler

In der Ereignisspur finden sich nach der Messung Informationswerte der Form:

- SYNC-Ereignis bei Rahmenende (Ereignis enthält SYNC-Kennung mit RSR-Daten),
- Ereignis aufgrund eines erfolgreichen Transfers (Ereignis enthält RSR-Daten sowie die Sende-/Empfangskennung) oder
- beide Ereignisse (erstes Ereignis mit SYNC-Kennung und RSR-Daten, zweites Ereignis mit Sende-/Empfangskennung und denselben RSR-Daten).

So kann der Sonderfall eintreten, daß zwei Ereignisse nach einem Rahmen übergeben werden: der vorherige Rahmen war für das betrachtete Objektsystem bestimmt und im folgenden Rahmen ist das SYNC-Muster enthalten. In diesem Fall wird bis T_{K6} nach Rahmenende der erfolgreiche Transfer des vorherigen Rahmens gemeldet, bevor im nächsten Takt (T_{K7}) das SYNC-Ereignis des aktuellen Rahmens signalisiert wird.

Lädt man als Vergleichsmuster die MAC-Adresse des Objektsystems in den VLSI-Mustervergleicher, so kann immer bei einem Sende- oder Empfangstransfer ein SYNC-Ereignis allein zum Erfassen des Rahmenendes erzwungen werden, dem bei erfolgreichem Sende- oder Empfangsvorgang ein zweiter Eintrag mit identischen Werten des RSR nachfolgt.

6.1.1.2 Systembus

Bei den verwendeten Speicherzugriffen stehen die geschriebenen Werte mit Ende des Buszyklus im FIFO-Speicher: bei beiden Rechnern dauert ein Schreibzyklus signifikant länger als das Einschreiben in den FIFO-Speicher, so daß mit Ende des Schreibzyklus die Meßdaten in den FIFO-Speicher eingeschrieben sind.

Die Verzögerungszeit bis zur frühest möglichen Abfrage aus dem FIFO-Baustein ergibt sich unter Verwendung von (Gl.17) zu:

$$T_{Bus} = T_D \qquad \text{(Gl.20)}$$

In allen beobachteten Objektsystemen werden mit denselben Meßfühlern Informationen extrahiert, weshalb die gemeinsame FIFO-interne Verzögerungszeit T_D in (Gl. 18), (Gl.19) und (Gl.20) gestrichen werden darf (entspricht einer Verschiebung der Zeitachse um T_D).

Damit ergibt sich, daß die Meßdaten, die über den Meßfühler am Systembus übergeben werden, schneller erfaßt werden als die durch aufwendige Hardware-Meßfühler erkannten Ereignisse direkt am Netzzugang. Wartende Meßdaten im FIFO-Bauelement sind nur dann möglich, wenn der Abfragezyklus größer als die Zeit zur Generierung neuer Informationen ist. Bei allen durchgeführten Messungen wurde der schnellstmögliche Abfragezyklus (1.28 µs) gewählt, der um den Faktor 5 kleiner als der minimale Generierungsabstand (6 µs) ist.

6.1.2 Erzeugung der Meßinformation durch Meßstatements

Die meisten Programme sind im LARGE-Speichermodell kompiliert. Damit muß bei Zugriffen auf Speicherzellen, die nicht im gerade aktuellen Datensegment liegen, ein Segmentregister neu geladen werden. Diese Aktion führt der Assemblerbefehl *les* (load pointer to register ES) in den folgenden Kodebeispielen aus. Dazu sind folgende einfache Sequenzen zur Wertbildung samt dem folgenden Schreibzugriff gemessen worden:

```
*ZeigerAufMonitorRegister = 5;                  /* fixer Wert */
*ZeigerAufMonitorRegister = i;                  /* variabler Wert */
*ZeigerAufMonitorRegister = proc | parameter;   /* Wert ODER-verknüpft */
```

In Tabelle 17 wird beispielhaft die vom C-Compiler unter XENIX erzeugte Assemblerkodesequenz angegeben.

Operand	Assemblerkode	Takte nach [91]
fix	mov es,$T2000	19
	les bx,DWORD PTR es:_ZeigerAufMonitorRegister	21
	mov WORD PTR es:[bx],205	3
variabel	mov es,$T2000	19
	les bx,DWORD PTR es:_ZeigerAufMonitorRegister	21
	mov ax,[bp-4]	5
	mov es:[bx],ax	5
ODER-verknüpft	mov es,$T2000	19
	les bx,DWORD PTR es:_ZeigerAufMonitorRegister	21
	mov ax,[bp-6]	5
	or ax,[bp-8]	7
	mov es:[bx],ax	5

Tabelle 17: Maximale Taktperioden für Operationen im LARGE-Modell unter XENIX

Für die auf der WS20 ablaufenden Betriebssysteme RTX286 und XENIX wurden gemittelte Ausführungszeiten entsprechend Tabelle 18 gemessen.

System	Sprache (Compiler)	Operation	Mittelwert (µs) (Datenblatt/gemessen)
RTX286	C (iC-286)	fix	7.5 / 4.5
		variabel	8.33 / 5.12
		ODER-verknüpft	9.49 / 5.83
XENIX	C (cc)	fix	7.16 / 6.14
		variabel	8.33 / 6.3
		ODER-verknüpft	9.49 / 7.5

Tabelle 18: Laufzeitmessungen im LARGE-Speichermodell

Die Differenz zwischen Datenblattangaben und gemessenen Werten kommt durch unterschiedliche Einflüsse zustande:

- infolge des Buszugriffs auf dynamischen Speicher sind z.T. mehr Taktperioden erforderlich als im Datenblatt [91] angegeben,
- der i80286 speichert intern einige Bytes, die bei erneuter Verwendung nicht mehr aus dem Speicher geholt werden müssen (Prefetch-Queue),
- beide Betriebssysteme operieren im Mehrprogramm-Betrieb und unterbrechen die laufenden Programme periodisch, was auch innerhalb der angegebenen Assembler-Sequenzen erlaubt ist und
- jeder Compiler verfügt über unterschiedliche Optimierungsstrategien, so daß das Segmentregister ES über den aufwendigen Befehl *les* nur dann nachgeladen werden muß, wenn dessen Inhalt zwischenzeitlich überschrieben wurde.

Da die einzelnen Meßstatements im instrumentierten Programm verstreut sind, dürften die gemessenen Werte kaum erreichbar sein [vergl. 167].

6.2 Meßgenauigkeit

6.2.1 Uhrengenauigkeit

Als Oszillatoren können in der Meßspeicherkarte verschiedene Quarzoszillatoren eingesetzt werden. Die Angaben in Tabelle 19 sind auf das Verhältnis der realtiven Frequenzstabilität $\Delta F/F$ bezogen.

Typ	Frequenzstabilität
unkompensiert (PXO)	±5ppm (Temperatur) ±2ppm (Spannung) ±1ppm (Alterung)
temperaturkompensiert (TCXO)	±2ppm (Temperatur) ±0.1ppm (Spannung) ±1ppm/Jahr (Alterung)
temperaturstabilisiert (OCXO)	±0.1ppm (Temperatur) ±10-8 (Spannung) ±10-6 /Jahr (Alterung)

Tabelle 19: Frequenzstabilität ($\Delta F/F$) der unterschiedlichen Oszillatortypen

Wenn eine Meßzeitabweichung von ΔZ_i toleriert wird, ergibt sich bei vorgegebener Frequenzstabilität nach Tabelle 19 die maximale Anzahl von Takten m der Dauer $t = 1/F$, nach der sich die Frequenzabweichung zu ΔZ_i aufsummiert hat.

$$m \leq \frac{\Delta Z_i}{t^*\left(\frac{\Delta F}{F}\right)} \qquad \text{(Gl.21a)}$$

Daraus ergibt sich dann die minimale Resynchronisationsrate R_Z:

$$R_z = \frac{F}{m} \geq \frac{\left(\frac{\Delta F}{F}\right)}{\Delta Z_i} \qquad \text{(Gl.21b)}$$

Für die drei oben genannten Oszillatortypen errechnen sich nach Addition aller Frequenzabweichungen mit (Gl.21b) für R_z die in Tabelle 20 angegebenen Werte. Gemäß dieser Tabelle genügt es zur Einhaltung einer Genauigkeitsgrenze von 10µs, daß spätestens alle T_{SYNC}=32.25 ms ein Resynchronisationspunkt in der Ereignisspur vermerkt wird, wenn ein Oszillator vom Typ TCXO (temperaturstabilisiert) verwendet wird. Bei Einsatz eines temperaturkompensierten Quarzes vom Typ OCXO ist bei einem Resynchronisationspunkt alle T_{SYNC}=100 ms eine zeitliche Genauigkeit von 1 µs erzeilbar.

	ppm	ΔZ_i=100µs	ΔZ_i=20µs	ΔZ_i=10µs	ΔZ_i=1µs
PXO	8	8	40	80	800
TCXO	3.1	3.1	15.5	31	310
OCXO	0.1	0.1	0.50	1	10

Tabelle 20: Notwendige Resynchronisationsrate R_Z in Abhängigkeit vom Oszillatortyp und der erlaubten Abweichung ΔZ_i

6.2.2 Synchronisation mittels Netzverkehr

Werden nur solche Rahmen zur Resynchronisation herangezogen, die durch die vorhandene Nutzkommunikation immer auf dem Netz sichtbar sind, so kann auf weiteren synthetisch erzeugten Verkehr verzichtet werden. Damit die in Kap. 6.2.1 abgeleitete Anforderung an die geforderte zeitliche Genauigkeit erfüllt wird, müssen die Trägerrahmen der SYNC-Kennung hinreichend oft, d.h mindestens alle T_{SYNC}, wiederholt werden. Rahmen mit solch regelmäßiger Häufigkeit sind nur bei der Kommunikation mit zentralen Betriebsmitteln wie Fileservern oder Printservern beobachtbar. Selbst in diesen Fällen kann es infolge des passiven Mitbeobachtens dieser Trägerrahmen vorkommen, daß die Zeitintervalle ohne das Auftreten eines Trägerrahmens bleiben. Während dieser Zeit sind somit keine SYNC-Ereignisse für die spätere Resynchronisation vorhanden. Das Korrelationsprogramm kann jedoch nachträglich virtuelle SYNCs eingefügen (vergl. Kap. 5.5).

In Bild 6.2a ist der Fall eingezeichnet, wo ein SYNC-Ereignis infolge Frequenzschwankungen außerhalb des gültigen Bereichs liegt. Dieser Bereich bestimmt sich aus einem Toleranzdreieck, dessen Grundseite $2*\Delta Z$ lang ist und dessen Mittelsenkrechte (Länge T_{SYNC}) zu der Grundseite gleichzeitig die Tangente an die Kurve beim letzten SYNC-Ereignis darstellt.

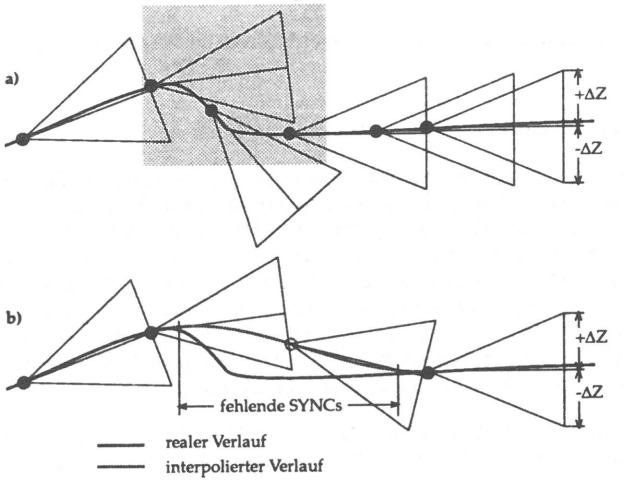


Bild 6.2: Toleranzbereiche der SYNC-Zeiten: a) SYNC liegt außerhalb, b) Einbau eines virtuellen SYNC-Ereignisses bei zu großem Abstand oder SYNC-Ereignisverlust

Bild 6.2b zeigt dieselbe Kurve, für die durch Interpolation zusätzlich ein SYNC erzeugt wurde, sowie die Toleranzdreiecke für diesen Fall. Wird der interpolierte Kurvenabschnitt wie im Bild eingefügt, so liegt das SYNC-Ereignis innerhalb des Toleranzdreiecks, da die reale Frequenzschwankung an den Randstellen nicht erkennbar war. Interpolation mit AKIMA-Verfahren erzeugt glatte Kurvenverläufe, während SPLINE-Interpolation Überschwinger und damit Verletzungen der Toleranzbereiche fördert [117].

Für die Überprüfung dieses Verfahrens wurde eine Variante untersucht, bei der unter Ausnutzung der Inactivity Control des Transportprotokolls ISO TP4 [97] der periodische Austausch von Quittungspaketen zur Überprüfung einer bestehenden Schicht 4-Verbindung als Auslöser der SYNC-Ereignisse dient. Die Erkennung erfolgt mit dem 48bit Mustervergleicher, der sechs aufeinanderfolgende Octetts (siehe Bild 6.3) in der Quittungs-PDU (extended ACK TPDU) vergleicht. Durch die fixe Länge im Mustervergleicher sind neben der TPDU-Identifikation (60_H) das Feld der Verbindungskennung (engl. Destination Reference) und ein Teil aus dem Feld zur Markierung des unteren Fensterendes (YR-TU-NR) erforderlich. Die Verbindungskennung wird beim Öffnen der Schicht 4-Verbindung dynamisch vergeben und ist somit für jede Messung unterschiedlich und nur den beteiligten Transportinstanzen bekannt. Wird diese Verbindung nur für die SYNC-Ereignisauslösung mißbraucht, so wird das Sendefenster nie geöffnet und im Feld YR-TU-NR bleibt immer der Wert 0. Für die beteiligten Stationen selbst ergeben sich keine Bearbeitungsschritte, da die Inactivity Control-Funktionalität für die Benutzer des Transportsystems unsichtbar bleibt.

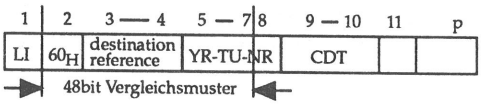


Bild 6.3: Extended ACK TPDU mit Bereich für 48 bit Vergleichsmuster

6.2.3 Synchronisation mittels synthetischem Verkehr

Ein Problem bei der Benutzung der Inactivity Control-Funktion besteht darin, daß die ausgelösten SYNC-Ereignisse aufgrund ihrer periodischen Übertragung sich schlecht für die Spurzuordnung durch Korrelation eignen: die Abstände zwischen den SYNC-Ereignissen sind zu ähnlich. Abhilfe schafft die zufällige Erzeugung von Netzpaketen an eine nicht vorhandene Zielstation. Da kein Empfänger existiert, kann ein beliebiges 48 bit Muster innerhalb des Rahmens zur Erkennung durch den 48bit Mustervergleicher vorgegeben werden. Zweckmäßigerweise wird dieser synthetische Rahmenstrom, der im Gegensatz zum Paketstrom bei der Inactivity Control-Funktion nur einen Sender verlangt, von einer Station am Segmentende erzeugt. Nun lassen sich die Laufzeiten zu den einzelnen Stationen durch ihre bekannte Busposition errechnen.

Die erzeugende Station sendet Rahmen aus, deren exakte Transferzeit sich nicht genau vorgeben läßt, da der Belegungszustand des Netzes die Übertragung jedes Rahmens beeinflußt. Gemäß CSMA/CD-Protokoll wird ein kollidierter Rahmen bis zu 16 mal wiederholt, bevor die Übertragung dieses Rahmens abgebrochen wird. Die mögliche Verzögerung nach einer Rahmenkollision errechnet sich aus:

$$T_V^n = \text{random}\left(0, \min\left(2^n, 2^{10}\right)\right) * T_{slot} \qquad \text{(Gl.22)}$$

wobei: n Anzahl bereits erlittener Kollisionen
T_{slot} Zeit für doppelte Kanallaufzeit (51,2 µs)

Zu dieser rahmenspezifischen Verzögerungszeit addiert sich die bereits verbrauchte Übertragungszeit $T_{ü}$ sowie die Kollisionserkennungszeit (T_{JAM}=3,2 µs) und die minimale Wartezeit zwischen aufeinanderfolgenden Rahmen (T_{GAP}=9,6 µs). Die gesamte Verzögerungszeit summiert sich zu:

$$T_V(k) = T_V^0 + \sum_{n=1}^{k} T_{ü}^{n-1} + T_{JAM} + T_{GAP} + T_V^n \qquad \text{(Gl.23)}$$

Gemäß dieser Gleichung erreicht die Gesamtverzögerung den Wert für den Toleranzbereich von T_{SYNC} nach Kap. 6.2.1 erst ab k=9 Wiederholungen, wenn die Übertragungszeit je kollidiertem Rahmen minimal (Null) ist.

Die Verzögerungen bei belegtem Kanal und nach Kollisionen verrauschen das Sendemuster, so daß ein zufälligeres Muster in Abhängigkeit des momentanen Netzverkehrs entsteht. Als sendende Station kann jede Station am Netz eingesetzt werden; so eignet sich auch die Steuerungsbaugruppe der Monitorkomponente zur Erzeugung der SYNC-auslösenden Rahmen.

Wird der Sendezeitpunkt eines Rahmens in der erzeugenden Steuerungsbaugruppe direkt in einer SYNC-Spur gespeichert, so erhält man damit eine Referenzspur, die zur Kontrolle und Zuordnung der anderen Ereignisspuren verwandt werden kann. An dieser Stelle würde sich auch der Einsatz einer extrem genauen Zeitbasis [z.B. 80] anbieten. Die Verwendung der SYNC-Spur als Referenzspur ist im Programm für Korrelation und Sortierung vorgesehen.

6.3 Rechengenauigkeit

Unvermeidliche Fehler treten bei der Darstellungskonvertierung von vorzeichenlosen 32 bit Zeitstempelformaten in 32 bit Gleitkommawerte auf.

6.3.1 Berechnungsgenauigkeit bei der Zeitsynchronisation

Bei der Bildung der globalen Ereignisspur treten Unterschiede in den Endergebnissen auf, wenn anstelle von XENIX286 das neuere UNIX386 als Plattform verwendet wird. Die Abweichung bei der Korrelationsrechnung trat in der sechsten Nachkommastelle bei Gleitkommazahlen auf und betrug eine Stelle. Daraus resultierende unterschiedliche Spurzuordnungen wurden bei den Tests nicht festgestellt.

Während der Bildung der globalen Ereignisspur wird der Kurvenverlauf zwischen den SYNC-Ereignissen interpoliert. Die dabei entstehenden Interpolationsfehler sind abhängig von der Interpolationsart (lineare, Spline- oder Akima-Interpolation). Jedoch ist all diesen Interpolationsarten gemein, daß die Stützstellen (SYNCs) Kurvenpunkte sind.

6.3.2 Berechnungsgenauigkeit im Statistikinterpreter

Das zur Quantilwertberechnung verwendete heuristische Verfahren von [108] liefert dann sehr genaue Schätzwerte (mittleres Fehlerquadrat « 0.02), wenn die Stichprobenanzahl mehr als einhundert Ereigniswerte umfaßt und die Klassenverteilung abschnittsweise einen parabolischen Verlauf annimmt.

Den Vergleich der intern mit schnellerer Ganzzahl-Arithmetik berechneten Klassenverteilung mit der exakten Klassenverteilung zeigt die Tabelle 21. Durch die unterschiedliche Klasseneinteilung infolge der Ganzzahlberechnung ergeben sich abweichende Klassenschranken und -häufigkeiten.

Klassenverteilung (log2)		Klassenverteilung (exakt log2)	
obere Grenze	Anzahl	obere Grenze	Anzahl
7.6928E-04	3	7.6928E-04	3
7.9744E-04	142	7.8455E-04	142
8.2560E-04	0	7.9983E-04	0
8.8192E-04	0	8.3037E-04	0
9.9456E-04	**33**	**8.9147E-04**	**2**
1.2198E-03	**0**	**1.0137E-03**	**31**
1.6704E-03	0	1.2580E-03	0
2.5715E-03	0	1.7468E-03	0
4.3738E-03	0	2.7243E-03	0
7.9782E-03	0	4.6793E-03	0
1.5187E-02	0	8.5893E-03	0
2.9605E-02	0	1.6409E-02	0
5.8441E-02	0	3.2049E-02	0
1.1611E-01	0	6.3329E-02	0
2.3146E-01	0	1.2589E-01	0
4.6214E-01	0	2.5101E-01	0
9.2352E-01	0	5.0125E-01	0
1.8463E+00	339	1.0017E+00	339

Tabelle 21: Vergleich einer mit Ganzzahlarithmetik berechneten Klassenverteilung mit der exakten Klassenverteilung

6.4 Meßdatenvolumen

6.4.1 Datenvolumen aufgrund einer Messung

Beim Auslesen der Meßdaten gibt es zwei Möglichkeiten:

- Gesamten Meßspeicher von n*32K Ereignissen auslesen (n=1, 3, 5, 7) oder
- Auslesen, bis eine Endekennung in einem Informationsfeld erkannt wird.

Im ersten Fall ergeben sich Dateien mit einer Größe von n*262144 Bytes für jede beteiligte Monitorkomponente. Wird ein vordefinierter Informationswert als Initialisierungswert für den Meßspeicher reserviert, so wird nur bis zu diesem Wert der Meßspeicher ausgelesen. Tritt dieser Informationswert als gültiges Feinmuster auf, so wird der Meßspeicher nicht bis zu seinem Ende ausgelesen, d.h. Meßdaten gehen verloren.

Die im Meßspeicher abgelegten 56bit breiten Ereignisse werden als vier 16bit Werte ausgelesen und zur Auswertestation transferiert. Damit erhöht sich die Meßdatenmenge im Verhältnis 7:8, schafft aber gleichzeitig Raum für die endgültige Darstellungsform in der globalen Ereignisspur.

6.4.2 Datenvolumen für Statistikauswertung

Für die Auswertung ist infolge der Trennung von Meßdatendatei und Referenzdatenbank der binäre Datensatz nur einmal notwendig, während für jede neue Auswertung eine Datenbeschreibung und Referenzdatenbank erforderlich wird.

Der für die Referenzdatenbank benötigte Speicherplatz läßt sich abschätzen durch:

- Datenbeschreibungsdatei
- Referenzdatenbank
  - Referenzdatei je Feinmuster der Datenbeschreibung mit Positionsangaben
  - Datei mit beiden Masken für jedes Feinmuster zur Mustererkennung
  - Datei mit Initialisierungswerten für jede Monitorkomponente

und beträgt bei einer Spurlänge von 32K Einträgen typisch 0,25 MBytes Plattenplatz, der zu der Ereignisspur von 0,25 MBytes dazugerechnet werden muß.

Die zur Auswertung verwendeten Programme beschränken die Datenmenge sehr stark, da aufgrund prozessorspezifischer Eigenheiten [91] nur bis zu einer Ereignismenge von 16000 je Feinmuster diese effizient implementierbar waren. Größere Ereignismengen sind durch abgewandelte Programme mit deutlich höherer Laufzeit bearbeitbar. Abhängig von den geforderten Optionen (Modalwert- oder Histogrammberechnung) wird diese Grenze bereits früher erreicht.

6.5 Rechenzeiten

6.5.1 Meßdauer

Die Dauer der Messung hängt von der Erzeugungsrate in einem Objektsystem ab. Im verteilten Fall wird die Messung beendet, sobald in einer Meßkomponente das Meßspeicherende erreicht wurde.

Die minimale Dauer einer Messung M hängt neben der Anzahl datenerzeugender Meßfühler a auch von der Speicherlänge L und der Zykluslänge Z ab.

$$M = \frac{160ns * f * Z * L}{a} \qquad \text{(Gl.24)}$$

Dabei gilt:

- f Frequenzteiler mit $f \in (2, 8, 32, 64)$
- Z Zykluslänge mit $Z \in (4, 8)$

L Einträge im gesamten Meßspeicher mit L ∈ (32K, 96K, 160K, 224K)

a Anzahl aktiver Meßfühler je Zyklus (0 < a ≤ Z)

Die maximale Meßdatenrate R_1 je Meßfühler ergibt sich dann, wenn je Abfragezyklus dieser Meßfühler einen Informationswert am Abfragebus bereitstellen kann:

$$R_1 = \frac{1}{160ns * f * Z} \qquad \text{(Gl.25)}$$

Sind entsprechend Bild 4.8 mehr Meßfühler am Abfragebus angeschlossen, so vervielfacht sich dieser Wert, wie in Tabelle 22 gezeigt.

aktive Meßfühler (*a* von *Z*)	min. Meßdauer *M* [s] (32K / 96K / 160K Einträge)	max. Datenrate bei *a* aktiven Meßfühlern [1/s]
1 von 4	0,0419 / 0,1258 / 0,2095	781250
4 von 4	0,0104 / 0,0314 / 0,0523	3125000
1 von 8	0,0838 / 0,2516 / 0,4193	390625
8 von 8	0,0209 / 0,0629 / 0,1048	3125000

Tabelle 22: Minimale Meßdauern und maximale Meßdatenrate (Frequenzteiler f=2)

6.5.1 Transferdauer

Zum Transfer besteht die Möglichkeit, über eine serielle V.24-Leitung oder das CSMA/CD-Netz die Daten abzufragen.

Beim Transfer über die serielle V.24 Leitung werden kleinere Meßdatenblöcke (32 Bytes) verwendet als beim LAN-Transfer (512 Bytes). Bei einer möglichen Transferrate von 19200 Baud ergeben sich typ. 800 bits/s netto, d.h. 12,5 Ereigniseinträge/s. Diese niedrige Übertragungsrate wird durch die notwendige Abbildung nichttransferierbarer Zeichen, die Prüfsummenüberprüfung und den Handshake-Betrieb verursacht. Bei der LAN-Übertragung entfällt die eigene Prüfsummenkontrolle, es wird aber trotz schnellerem Bittakt nur ein Durchsatz von wenigen KBytes/s erreicht.

6.5.2 Korrelationsberechnungen

Zur Spurenzuordnung werden die Korrelationsberechnungen in parametergesteuerten Schleifen durchgeführt. Als Vorgabe für die Parametertripel dienen die Größen *S* (statisches Intervall) und *D* (dynamisches Intervall) aus Tabelle 23.

Parameter	Startwert	Endwert	Inkrement
S (Start)	S_S	S_e	S_i
S (Länge)	LS_S	LS_e	LS_i
D (Start)	D_S	D_e	D_i
D (Länge)	LD_S	LD_e	LD_i

Tabelle 23: Tripel aus Start- und Endewert sowie Inkrement zur Steuerung der Korrelationsintervalle

Die Anzahl durchgeführter Korrelationsberechnungen K berechnet sich dann zu:

$$K = \left\lfloor \frac{(S_e - S_s + 1)}{S_i} \right\rfloor * \left\lfloor \frac{(LS_e - LS_s + 1)}{LS_i} \right\rfloor * \left\lfloor \frac{(D_e - D_s + 1)}{D_i} \right\rfloor * \left\lfloor \frac{(LD_e - LD_s + 1)}{LD_i} \right\rfloor \qquad \text{(Gl.25)}$$

Zu beachten ist, daß mit jeder Berechnung bei inkrementierter dynamischer Intervallänge LD die Stützstellenanzahl wächst. Typische Vorgaben sind in Tabelle 24 als Fälle A..E eingetragen. Der Fall F stellt den Fall dar, bei dem der 100%-ig erkannte Spurenversatz bis zum 4-fachen typischen SYNC-Abstand (ca. 0,25 Sekunden) betragen kann.

	A	B	C	D	E	F
$S_a\ S_e\ S_i$	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1
$LS_a\ LS_e\ LS_i$	5 7 1	5 8 1	4 7 1	4 6 1	6 9 1	14 14 1
$D_a\ D_e\ D_i$	1 4 1	1 4 1	1 3 1	1 3 1	1 3 1	1 8 1
$LD_a\ LD_e\ LD_i$	3 3 1	3 4 1	3 3 1	3 5 1	3 5 1	4 4 1
je LD	12	16	12	9	12	8
insgesamt	12	32	12	27	36	8

Tabelle 24: Anzahl benötigter Korrelationsberechnungen für Fälle A..F

Die Zeitstempelwerte von drei generierten Ereignisspuren (gleichverteilte SYNC-Abstände von 50..100 ms) wurden mit normalverteiltem Rauschen mit Mittelwert 1 ms und Varianz 10^{-4} s^2 verrauscht. Bild 6.4 gibt die Anzahl richtig erkannter Verschiebungen in Abhängigkeit der Zufallszahlengeneratoren, der Intervallwerte aus Tabelle 24 sowie einem Startversatz (Offset) in 100 ms-Schritten und drei Spuren an.

Die hierbei verwendeten Zufallsgeneratoren

rand()	Zufallsgenerator des XENIX-Betriebssystems [181],
L'Écuyer	überlagerte Zufallsgeneratoren (16 und 32bit Version) [122],
subtraktive Methode	nach [116] und
linear kongruente Methode	

liefern vergleichbare Resultate, wobei eine Vergrößerung der statischen Intervalle S bei gleichzeitiger Anpassung der dynamischen Intervalle D eine signifikante Verbesserung der korrekten Zuordnungswahrscheinlichkeit ergibt. Kleine dynamische Intervalle liefern unbefriedigerende Resultate, erlauben aber die Erkennung größerer Verschiebungen.

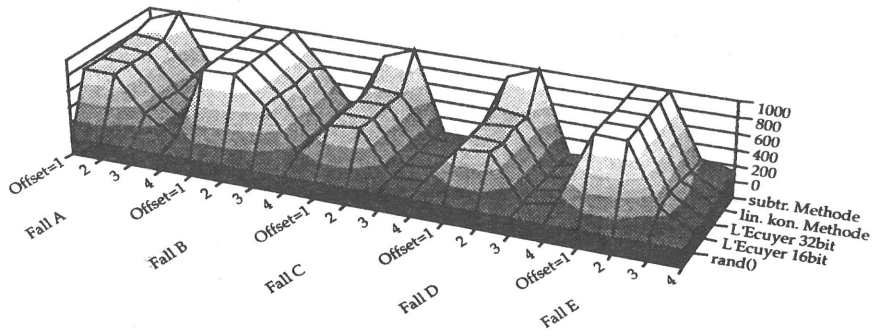


Bild 6.4: Korrelationsergebnisse bei simulierten Ereignisspuren (je 1000 Versuche)

Die Laufzeit für diese Spurzuordnungen betrug einschließlich der Ereignisspurerzeugung je Offset ca. 24 Stunden, woraus sich eine Rechendauer (i80286) von 17 s je Datensatz ableiten läßt.

6.5.3 Statistikauswertung

Die Berechnungsdauer im Parser wird stark beeinflußt von den Musterdefinitionen. Sind die zur Erkennung der Feinmuster notwendigen Masken bestimmt, wird für jedes Feinmuster die Ereignisspur vom Anfang an durchsucht. Es hat sich gezeigt, daß hierbei die Zugriffsgeschwindigkeit auf die Magnetplatte sowie die XENIX-internen Puffermöglichkeiten die Zeit bis zum Aufbau der Referenzdatenbank bestimmen. Da dieser Schritt je Ereignisspur nur einmal durchgeführt wird, wurden die Einflußfaktoren nicht näher untersucht.

Der Interpreter wurde mit den systembekannten Mitteln (*prof* [181]) genauer auf sein Laufzeitverhalten untersucht, da durch eine Abänderung der Typ- und Paardateien neue Auswertungen unterstützt werden. In [55] sind beispielhaft Ergebnisse der Laufzeitmessungen des Interpreterlaufs an-

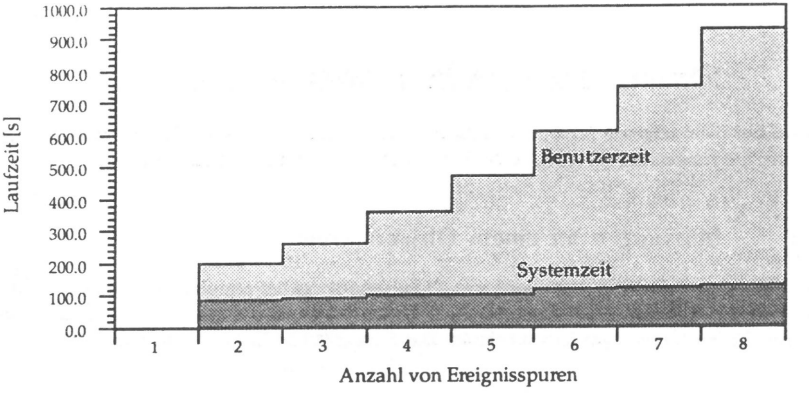


Bild 6.5: Rechenzeit für 1000 Korrelationsberechnungen (i80386 Prozessor) für Fall F

gegeben. Dabei zeigt sich, daß die Modalwertberechnung den Hauptanteil an der Laufzeit ausmacht, gefolgt von der Quantilberechnung (Bild 6.6).

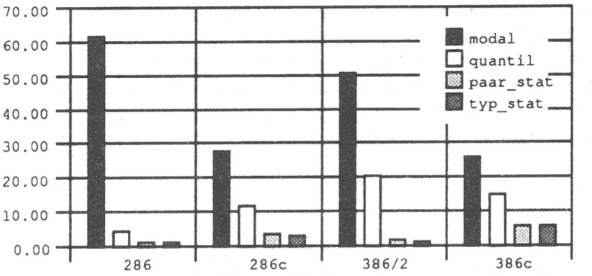


Bild 6.6: Prozentualer Laufzeitanteil bei 15K Ereignissen (prozessorspezifisch für i80286, i80286 mit Numerik-Koprozessor, i80386 aber i80286 Kode, i80386 mit Numerik-Koprozessor)

Wird die Datenmenge größer, so steigt die Gesamtbearbeitungszeit überproportional an, wie Tabelle 25 zeigt.

	kurze Spur	lange Spur	Verhältnis (lang:kurz)
Datenmenge	15536 Ereignisse	98303 Ereignisse	6.327
Laufzeit	28.3 s	469.28 s	16.582

Tabelle 25: Laufzeit in Abhängigkeit von der Spurlänge

7 Einsatz des verteilten Meßsystems

Einige Beispiele zeigen anhand durchgeführter Messungen an unterschiedlichen Stellen der Objektsysteme die Anwendbarkeit, aber auch die Grenzen, des verteilten Meßsystems.

7.1 Messungen an einem Objektsystem

Zur Messung an einem System wird eine Monitorkomponente mit einem Systembus-Meßfühler eingesetzt. Die zeitliche Auflösung beträgt 1.28 µs. Die Objektsysteme sind bei allen Messungen nur mit der jeweiligen Applikation belastet. Das Netz selbst ist Teil des gesamten Institutsnetzes.

7.1.1 IHI-Treiber auf WS20 unter RTX286

Zur Bewertung des IHI-Treibers unter dem Echtzeitbetriebssystem RTX286 [173] wurde der Treiber-Quellkode mit den Meßstatements ❶...❽ bestückt:

- ❶ Beginn der Übergabe eines Auftragspuffers an das Transportsystem,
- ❷ Eintrag des Auftrags in IHI-interne Referenztabelle,
- ❸ Signal an Subsystem zur Übernahme des Auftrags (durch DMA),
- ❹ Ende der Auftragsübergabe bei Applikation,
- ❺ Applikation beginnt auf abgearbeiteten Auftrag zu warten,
- ❻ empfangener Rahmen wird über ein Unterbrechungssignal dem IHI-Treiber bekanntgemacht,
- ❼ Rückübersetzung der Adressen und Zuordnen des Pakets an die Applikation und
- ❽ Applikation hat das Paket erhalten und beendet synchronen Lesezugriff.

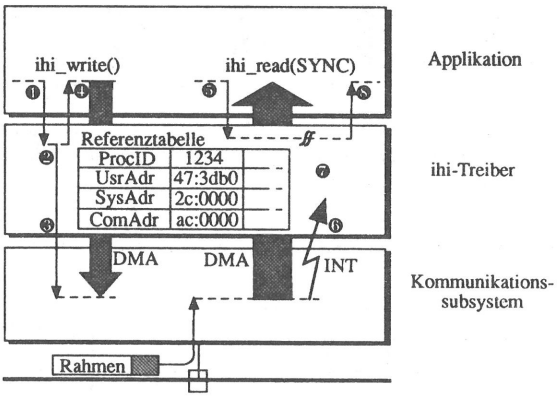


Bild 7.1: IHI-Treibermodell unter RTX286

Die in Bild 7.1 erkennbaren DMA-Transfers sind mit Meßstatements nicht markierbar, so daß nur die eingezeichneten 8 Punkte zu identifizieren sind. Die Übergabezeit des Auftrags an das Kommunikationssubsystem bei ❸ wurde durch zwei begrenzende Meßpunkte markiert, so daß die Zeit vom Anzeigen eines Auftrags bis zur Quittierung im Statusfeld des Auftrags innerhalb des Zeitintervalls T(❸) liegt.

Bedeutung	Anzahl	Minimalwert	Mittelwert	Maximalwert	Varianz
IHI-Treiber intern					
❷ -> ❸	968	0.4390	0.5803	1.02656	4.7860E-10
❸ -> ❻	968	1.6281	6.0728	22.8059	4.2108E-2
❻ -> ❼	968	1.2979	1.3832	3.4270	2.6080E-8
T(❸)	968	0.1950	0.2457	0.7475	2.9676E-9
Schreibaufträge (❶ -> ❹)					
T(ihi_write(OPEN))	4	4.6720	4.6963	4.7104	-
T(ihi_write(CR))	4	3.9641	4.0124	4.0921	-
T(ihi_write(VC))	957	3.8061	3.8824	7.8172	1.6427E-8
Leseaufträge (❺ -> ❽)					
T(ihi_read(OPEN))	4	3.3894	3.4192	3.4496	-
T(ihi_read(CR))	4	15.1398	1529.9470	6073.6060	-
T(ihi_read(VC))	957	10.4012	17.4746	1478.3050	4.302888E-3

Tabelle 26: Meßwerte nach Transfer von 4 Dateien über den RTX286 IHI-Treiber (gemessene Zeiten in ms)

Die gemessenen Zeiten innerhalb des IHI-Treibers sind nahezu konstant (Tabelle 26). Die Wartezeiten, die die Applikation an der IHI-Schnittstelle sieht, schwanken beim Übergeben eines Auftrags (❶->❹) kaum, jedoch beim Empfang der Aufträge (❺->❽) sehr stark.

7.1.2 CNMA-Gateway auf Intel310 unter iRMX II

Die folgenden Meßergebnisse zeigen den zeitlichen Pollingablauf im Gateway [170], das zwei Stationen im Pollingbetrieb abfrägt, um dem Netzwerkmanager Objekte aus nicht ISO-konformen Systemen verfügbar zu machen. Dabei wird nach jeder Systemabfrage eine vorgegebene Zeit (2 Sekunden in Bild 7.3) gewartet, bevor dieselbe Station erneut abgefragt wird. Je Station werden nacheinander unterschiedliche Objekte abgefragt, wobei manche Objekte leer bzw. lokal abgebildet (Phasen *PMOleer - PMOfertig*) sind und andere durch reale Abfragen (Phasen *ihiwrite* und *ihiread*) über das Netz erfragt werden müssen.

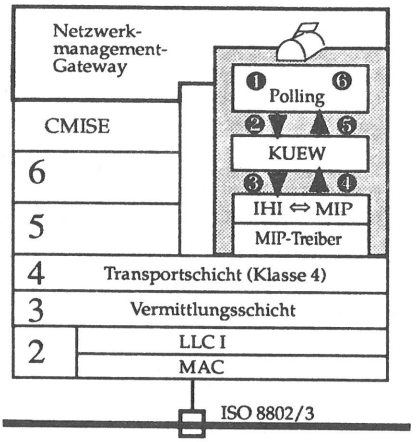


Bild 7.2: Schichtenmodell des CNMA-Gateways

Folgende Phasen treten dabei in Bild 7.3 und Bild 7.4 auf, vergl. Tabelle 27.

Phase	Name	Bedeutung
0	PStart	Beginn eines Pollingschrittes (❶)
1	PMOleer	Zugriff auf leeres (lokal abgebildetes) Objekt (❷)
2	PMOfertig	Zugriffsende (❺)
3	PMOlesen	Zugriff auf entferntes Objekt (❷)
4	ihiwritevor	Aufruf zur Übergabe einer SDU an Schicht 4 (❸)
5	ihiwritenach	Aufrufende (❸)
6	ihireadvor	Aufruf zum Lesen einer SDU von Schicht 4 (❹)
7	ihireadnach	Aufrufende (❹)
8	PMOAusw	Auswertung (❺)
9	PStop	Ende eines Pollingschrittes (❻)
10	PANachklapp	sofortiger Neubeginn eines Pollingschrittes (❶)
11	PArcvmsg	Warten auf neuen Pollingauftrag

Tabelle 27: Phasen beim Abfragen der Stationsobjekte (Managed Objects, MO)

Anstelle der statistischen Auswertung soll hier das Ablaufgeschehen anhand von Gantt-Diagrammen aufgezeigt werden. Die Erkennung der Phasen verlangt, daß zu jedem Phasenbeginn ein Feinmuster erzeugt wird. Der Parser analysiert die Ereignisspur und erstellt einen Datensatz, der als Eingabe für verfügbare grafische Auswertungsprogramme [z.B. 84] dient. Eine mögliche Anzeigeform ist das in Bild 7.3 gezeigte Gantt-Diagramm.

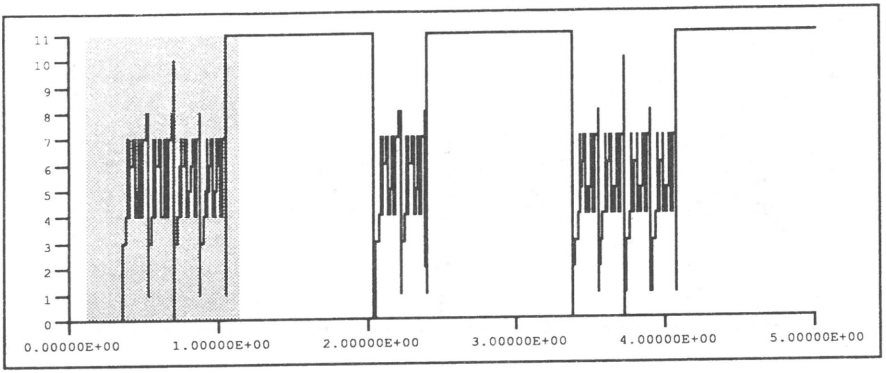


Bild 7.3: Polling zweier Stationen mit je 2 Sekunden Abstand

In Bild 7.3 erkennt man, daß zwischen den Ruhezeiten (je 1 Sekunde) ein Pollingschritt durchgeführt wurde. Nach dem ersten und dritten Pollingschritt ist zusätzlich ein weiterer Pollingschritt gestartet worden, da am Ende jedes Pollingschrittes geprüft wird, ob ohne Ruhezeit erneut eine Abfragerunde gestartet werden soll. Der in Bild 7.3 grau unterlegte Bereich wird nun in Bild 7.4 (folgende Seite) jeweils weiter vergrößert dargestellt.

Die periodische Abfrage der Objekte läßt unter Kenntnis der Zugriffsabfolge (mehrere lokale Objekte, dann Objekte von entfernten Systemen und wieder lokale Objekte) ein entsprechendes Diagramm mit zwei dominanten Abstandszeiten in der dynamischen Anzeige erwarten (Bild 7.5).

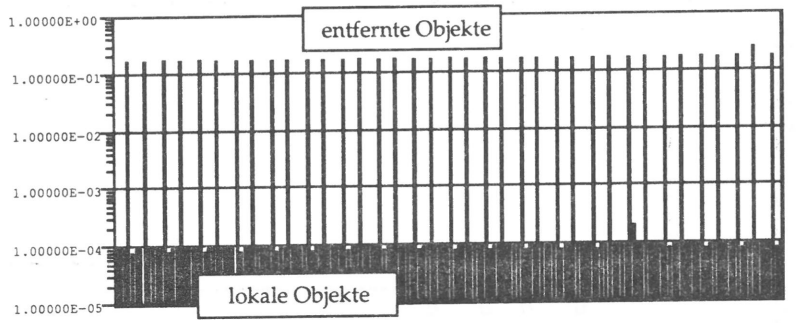


Bild 7.5: Dynamischer Verlauf der Zugriffszeiten auf Objekte (❷ bis ❺)

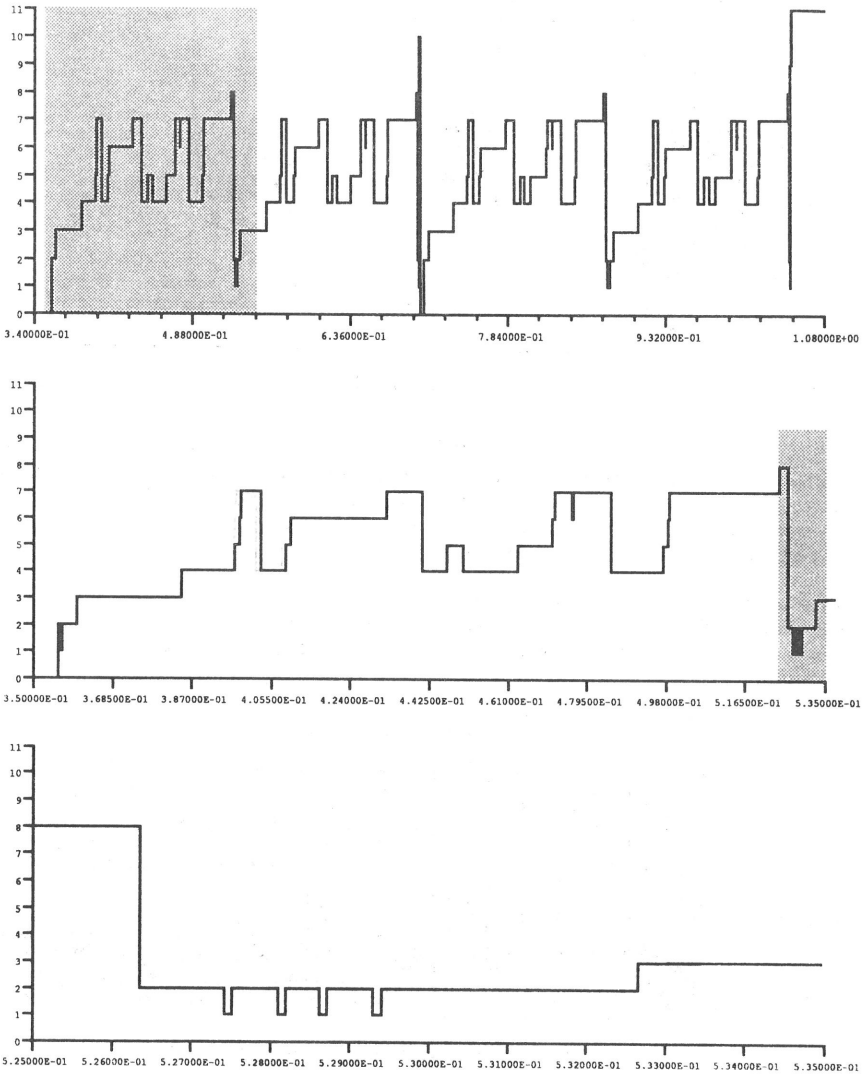
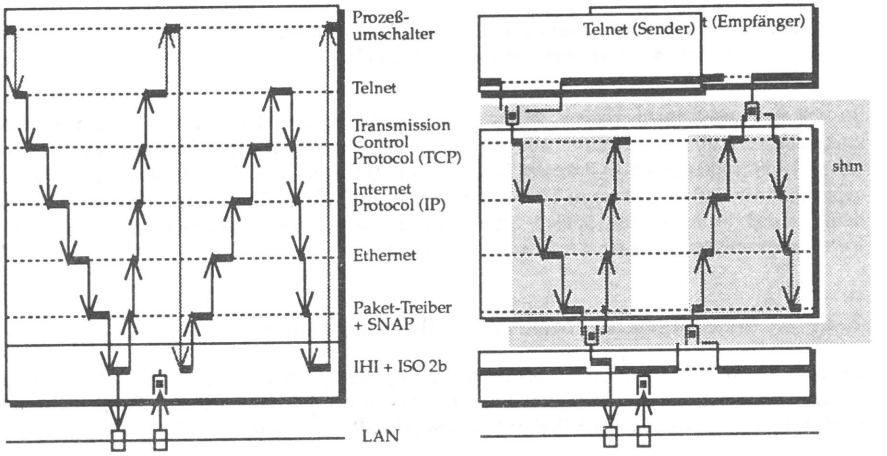


Bild 7.4: Schrittweise Vergrößerung: a) ein Pollingschritt mit sofortigem Pollingneubeginn, b) Abfrage lokaler (leerer) und entfernter Objekte, c) Abfrage lokaler Objekte

7.1.3 TELNET-Applikation

Als Beispiel einer Messung an einem Protokollstack über alle Schichten entsprechend dem Internet-Modell [26] dient eine Implementierung [56] des TELNET-Dienstes [198] zum interaktiven Zugang auf ein entferntes Rechensystem. Die ursprüngliche Einprozeßversion wickelte über einen intern realisierten Umschalter alle parallel ablauffähigen Sitzungen ab. Die neuere Version unterscheidet Prozesse für Applikationen und zur Abwicklung des Transportprotokolls. Die Synchronisation erfolgt über Signale und gemeinsame Speicherbereiche (engl. Shared Memories, shm). Zur Kommunikation mit entfernten Systemen über die verfügbare Netzzugangsbaugruppe [88, 89] ist zusätzlich das Sub-Network Access Protocol (SNAP) [160] erforderlich (Bild 7.6).



a) Einprozeßversion b) Mehrprozeßversion

Bild 7.6: Struktur der KA9Q Internet-Software unter XENIX (shm shared memory)

Wesentlicher Unterschied der beiden Versionen ist, daß in der Einprozeßversion [212] eine sequentielle Abarbeitung der Sende- bzw. Empfangspakete durchgeführt wird, während die Aufspaltung auf 4 Prozesse (vergl. Bild 7.6b) in der Mehrprozeßversion [177] eine quasi-parallele (asynchrone) Abarbeitung erlaubt. Für die Auswertung der Meßspuren bedeutet dies, daß zwar die „happened before"-Relation von [121] gilt, jedoch die einfache Zuordnung der Musterpaare, sofern sie zu unterschiedlichen Prozessen gehören, nicht immer gültig ist. Im folgenden werden deshalb nur exemplarische Angaben zur Typstatistik gemacht. In Tabelle 28 sind die Eintrittspunkte in einen der Prozesse (Telnet-Anwendung, TCP/IP samt SNAP-Protokollabwickler und Schnittstellentreiber (IHI)) nach Transferrichtung getrennt aufgelistet.

Bei der Multitasking-Implementierung werden nicht alle Daten aus den Netzrahmen unmittelbar an

Bedeutung	Anzahl	Minimalwert	Mittelwert	Maximalwert	Var.koeffizient
Empfangsrichtung					
Rahmen empfangen (ISO 2b)	269	0.01938	1.05336	18.8841	2.17200
Paket empfangen (IP)	268	0.04547	1.04691	18.8212	2.18739
Daten empfangen (Telnet)	152	0.16840	1.21774	11.6889	1.21591
Senderichtung					
Daten eingelesen (Telnet)	196	0.05289	0.92989	11.8394	1.7697
Paket zu senden (TCP)	201	0.03288	0.91775	11.6262	1.7306
Rahmen senden (ISO 2b)	451	0.01908	0.62100	18.7271	2.9481

Tabelle 28: Meßpunktabstände bei Telnet-Anwendung für ein Meßintervall von 4:43.315 min (gemessene Zeiten in ms)

die Telnet-Anwendung übergeben. Vielmehr wird der Telnet-Applikation das Vorliegen von Empfangsdaten signalisiert, woraufhin alle wartenden Daten aus dem Shared Memory gelesen werden (Anzahl von empfangenen IP-Paketen: 268, Telnet-Empfangsaufrufe: 152). In Senderichtung hingegen fällt auf, daß scheinbar pro TCP-Sendepaket zwei Treiberaufrufe gemacht werden. Ursache dafür ist, daß die verwendete Schicht 2-Software erst Netzrahmen empfangen kann, wenn zuvor leere Rahmenpuffer *gesendet*, d.h. übergeben wurden.

7.2 Messungen an mehreren Objektsystemen

Der auf einer WS20 unter XENIX ablaufende Sendeprozeß (SEND) (vergl. Bild 7.7) baut die ISO 4-Verbindung auf und überträgt alle Daten in Blöcken von jeweils 1 KByte. Das Transferende wird im Empfangsprozeß (RECEIVE) auf dem Intel310-Rechner unter iRMX II am Erhalt eines

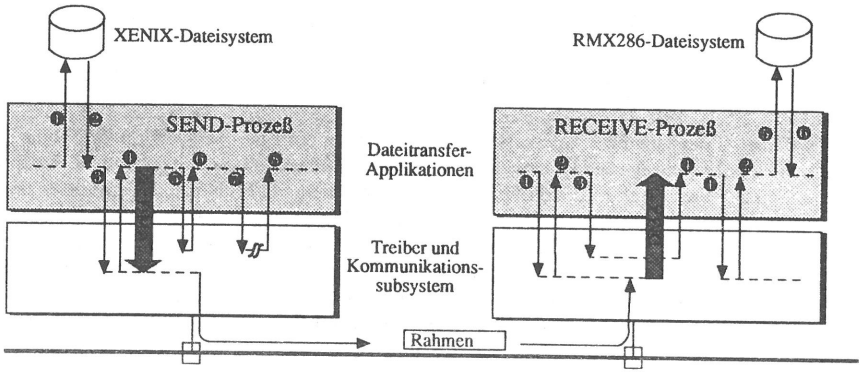


Bild 7.7: Modell eines Dateitransfers über die Transport-Schicht ISO TP4

kleineren Blockes erkannt, woraufhin er die Verbindung abbaut. SEND holt während der Transferphase einen Datenblock vom Hintergrundspeicher (❶, ❷) und übergibt diesen an das Trans-

portsystem (❸, ❹). Sind nur noch *smin* Sendepuffer frei, so wird geprüft, ob ein Sendepuffer zurückgelesen werden kann (❺, ❻). Erst wenn alle *smax* Sendepuffer belegt sind, wird auf die Rücknahme eines Sendepuffers vom Transportsystem gewartet (❼, ❽).

Im Empfangsprozeß wird ein Wechselpuffermechanismus verwendet. Zuerst wird bei (❶, ❷) ein Empfangspuffer an die Schicht 4 übergeben. Nun wird in (❸, ❹) auf die Rücknahme eines Puffers gewartet. Bevor dieser Puffer auf dem Hintergrundspeicher abgelegt wird (❼, ❽), wird der andere Puffer an das Transportsystem übergeben (❺, ❻). Dadurch kann parallel zur Plattenoperation ein nachfolgender Rahmen empfangen werden.

Die systemspezifischen Meßfühler werden über den Abfragebus gemeinsam an eine Monitorkomponente angeschlossen. Die Systembus-Meßfühler in den Objektsystemen werden dann alternierend abgefragt, die zeitliche Auflösung der Zeitstempel betrug dabei 2.56 µs.

7.2.1 Synchroner Filetransfer

Im synchronen Fall wird vom Sendeprogramm nur ein Puffer verwendet, d.h. $smax = smin = 1$. Der Sender muß dann bis zum Empfang der Schicht 4-Quittung warten, bevor der nächste Datenblock von der Platte gelesen und an die Transportschicht übergeben werden kann. Im Empfangsprozeß wird ebenfalls nur ein Puffer verwendet, d.h. erst muß der Datenblock auf Platte geschrieben werden, bevor das Objektsystem erneut empfangsbereit ist (❶❷❸❹❼❽→❶...).

Die Synchronität zwischen Sender und Empfänger erlaubt, daß im Statistikinterpreter die Musterpaare der lokalen Prozesse und das Paar SEND❸->REC❹ korrekt zugeordnet werden können, während die Zuordnung bei SEND❶->REC❻ aufgrund der Überlappung von Plattenzugriffen mit Netzaktionen nicht stimmt (siehe Bild 7.8 und Tabelle 29).

7.2.2 Asynchroner Filetransfer

Mit geänderten Werten für *smin=4* und *smax=1* kann der Verbindungsdurchsatz deutlich erhöht werden. Das maximale Sendefenster ist wie im synchronen Fall eins, jedoch speichert auf Sendeseite das Transportsystem maximal *smax* Puffer zwischen. Dadurch sind dort immer Datenblöcke sendebereit, so daß der Verbindungsdurchsatz nun durch die Verarbeitungsleistung im Empfangsprozeß RECEIVE begrenzt wird, obwohl zwei Empfangspuffer im Wechsel benützt werden.

Bild 7.9 zeigt, daß nun die strikte Reihenfolge von Pufferübergabe im Sender und Puffererhalt im Empfänger nicht mehr eingehalten wird. Die Zuordnung allein durch Markierung der Pufferübergabe bzw. Puffererhalt vom IHI-Treiber würde im Statitikinterpreter falsche Verzögerungszeiten liefern. Abhilfe schafft eine Identifikation der einzelnen pufferbezogenen Aktionen durch Angabe einer laufernden Kennnummer (*n* in Bild 7.9) als Teil des Informationsfeldes. Sind mehrere Sendeprozesse gleichzeitig aktiv, muß zusätzlich eine Prozeßidentifizierung im Informationsfeld (vergl. Bild 4.13) vorgesehen sein, die die verbleibende Informationsfeldbreite weiter einschränkt.

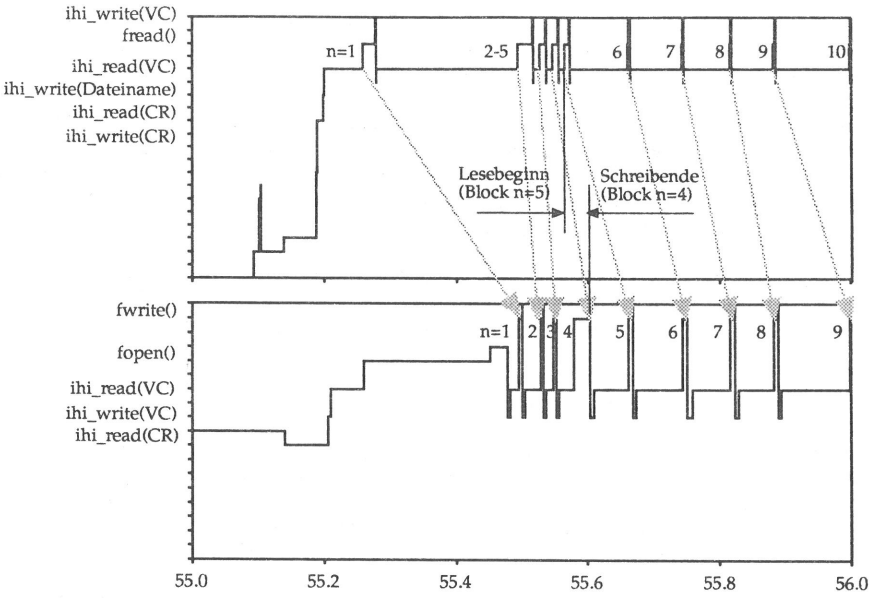


Bild 7.8: Synchroner Dateitransfer zwischen Sender (oben) und Empfänger (unten) (SEND❶->REC❻ graue Markierungen bei n=1,2...)

Bedeutung	Anzahl	Minimalwert	Mittelwert	Maximalwert	Varianz
SEND					
❶ -> ❷	257	0.6528	5.4581	88.7475	7.6004E-5
❷ -> ❼	257	2.6649	2.7724	3.8118	4.2355E-8
❼ -> ❽	259	17.3081	158.7004	429.0637	2.3813E-3
RECEIVE					
❶ -> ❹	256	27.6582	151.1372	218.7904	1.7259E-3
❹ -> ❺	256	0.0563	0.0601	0.18176	1.1740E-10
❺ -> ❻	257	0.5120	14.0300	48.1715	9.7380E-6
SEND-RECEIVE					
SEND❸ -> REC❹	250	0.19456	159.9341	434.8160	2.9697E-3
SEND❶ -> REC❻	254	**10.9030**	**17.3211**	**76.2368**	**1.77919E-5**
Abstand (SEND❸)	256	31.5238	166.9127	477.5014	2.2634E-3
Abstand (REC❹)	256	39.0016	165.7790	230.7072	1.6773E-3

Tabelle 29: Meßwerte für synchronen Dateitransfer (gemessene Zeiten in ms)

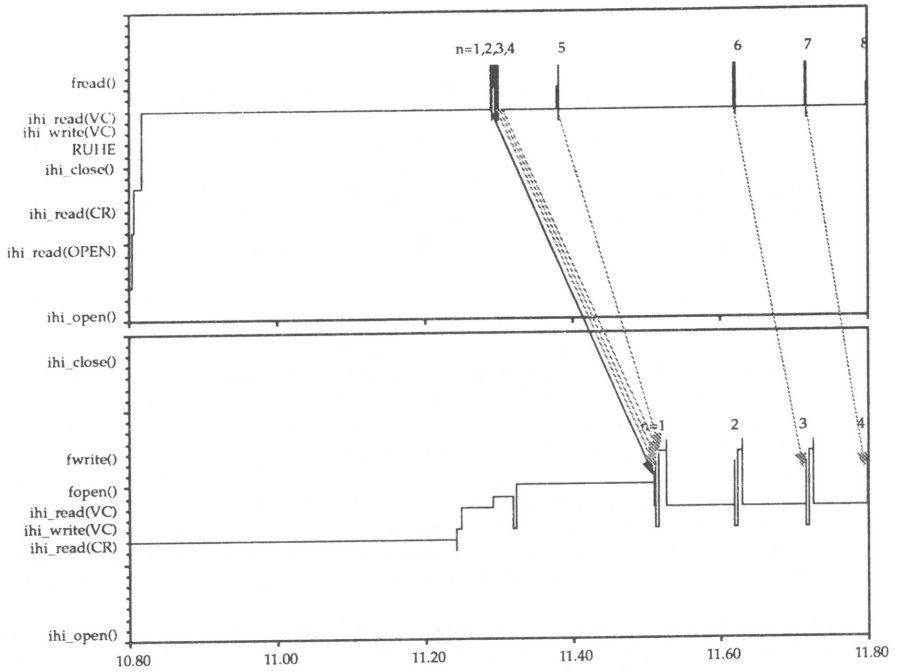


Bild 7.9: Transferbeginn beim Datentransfer mit asynchron laufenden Prozessen zwischen Sender (oben) und Empfänger (unten)

Eine Definition von Feinmustern über Grobmuster mit fortlaufender Kennung ist mit der Datenbeschreibungssprache über variable Felder möglich. Die Zuordnung der Feinmuster zu Musterpaaren wird im Statistikinterpreter jedoch noch nicht unterstützt.

8 Zusammenfassung und Schlußbemerkungen

Die Anforderungen an das verteilte Meßsystem umfaßten die Möglichkeit der gleichzeitigen Messung in allen Teilen eines verteilten Systems, das an einem Lokalen Netz (CSMA/CD) unter realen Bedingungen in Betrieb ist, sowie den Verzicht auf zusätzliche Verbindungsnetze zum Meßdatentransport oder zur Synchronisation. Anwendung sollte das verteilte Meßsystem bei der Messung Protokollstack-interner Abläufe in Steuerungs- und Zellrechnern in einer CIM-Umgebung finden.

Das gewählte Konzept basiert auf einem verteilten Hybridmonitor, der über dedizierte Meßfühler am Netzzugang und den Systembussen verfügt. Aktionen am Netzzugang werden dabei durch passive Hardware-Meßfühler erkannt, während alle in Software ablaufenden Aktionen durch den Einbau von zusätzlichen Schreibanweisungen auf Systembus-Meßfühler angezeigt werden. Alle erfaßten Meßdaten werden für die Meßdauer in dezentralen Monitorkomponenten zwischengespeichert und nach der Messung über das vorhandene Lokale Netz an eine zentrale Auswertestation übertragen. Die dezentralen Monitorkomponenten verfügen über nichtsynchronisierte Uhren, die Zeitstempel zur zeitlichen Markierung der Meßdaten bereitstellen. Das Synchronisationsproblem dieser frei laufenden Uhren kann gelöst werden, indem die Auftrittszeiten von Netzrahmen als Resynchronisationspunkte verwendet werden. Die statistische Auswertung der gemessenen Daten verlangt, daß zuvor mittels der Datenbeschreibungssprache DDL die Informationskodierung beschrieben und zu jedem Meßereignis ein Eintrag in eine Referenzdatenbank vorgenommen wird.

Für erste Messungen wurde ein Prototyp des verteilten Meßsystems aufgebaut. Als Objektsysteme standen Rechner mit Zugang zu einem LAN mit CSMA/CD-Zugriffsprotokoll und Kommunikationsprotokollen nach ISO 8802/2 bzw. ISO 8072 zur Verfügung. Mit den aufgebauten Meßkomponenten lassen sich Messungen in einzelnen oder verteilten Objektsystemen am Netzzugang und dem Systembus durchführen. Die Einsatzbereitschaft für derartige Messungen wurde mit den Beispielen aus Kap. 7 aufgezeigt.

Neben den Verzögerungen in den einzelnen Meßfühlern bestimmt die Uhrengenauigkeit die erzielbare Meßgenauigkeit. Abhängig von der Rate der betrachteten Netzrahmen ergibt sich eine Uhrengenauigkeit bis in den Bereich von 10 µs. Zu diesem Wert addiert sich die Verzögerung in den Meßfühlern am Netzzugang bzw. den Systembus-Meßfühlern. Außer bei den Meßfühlern, die den korrekten Transfer eines Rahmens am Netz erfassen, liegen die Verzögerungen im Bereich einer Mikrosekunde.

Daß am Ende dieser Arbeit Verbesserungsmöglichkeiten erkennbar sind, war aufgrund der konträren Anforderungen (Einfachheit vs. Genauigkeit) zu erwarten.

So stellt die Zeitzuordnung einen Problemkreis dar. Eine definiertere Zeitzuordnung wäre realisierbar, wenn der Zeitstempel direkt beim Erzeugen der Meßinformation zugeordnet würde und nicht erst nach der Entkopplung über das FIFO-Element. Damit wird aber der Abfragebus breiter:

- Signalleitungen zum Takten und Rücksetzen des Zählers in den Meßfühlern und
- Signalleitungen für den Zeitstempel von den Meßfühlern.

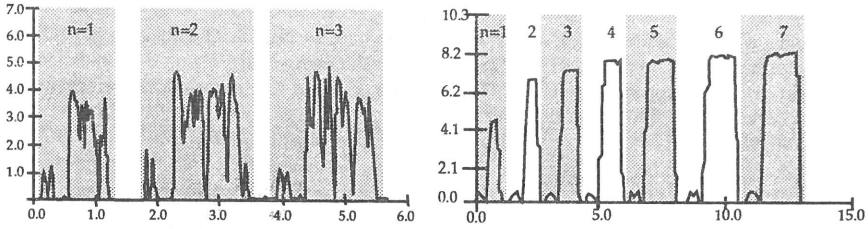
In einer Implementierung würde dies eine Aufweitung der bestehenden 28 auf 50 Signalleitungen zuzüglich der Abschirmungen bedeuten (dieser Bus wurde bewußt schmal gehalten, damit die Erweiterung auf differentielle Leitungspaare zur Überwindung auch größerer Entfernungen möglich bleibt). Zusätzlich ist dann in jedem Meßfühler ein Zeitzähler notwendig. Damit entsteht keine Unsicherheit bei wartenden Einträgen im FIFO-Speicher, und die Korrektur der Auftrittszeiten in unterschiedlichen Meßfühlern infolge spezifischer Verzögerungen kann entfallen. Für die Erkennung erfolgreich gesendeter bzw. empfangener Rahmen bedeutet dies, daß wie bei der SYNC-Mustererkennung das Rahmenende zeitlich markiert und bis zur endgültigen Entscheidung zwischengespeichert werden muß. Als Folge davon können Einträge in nicht geordneter Zeitreihenfolge in der Spur abgelegt sein, was durch eine nachfolgende Sortierung behoben werden könnte.

Der gleichzeitige Start einer Messung wird bislang vom Kommunikationsprotokoll durch die Übertragung eines Startbefehls in einem Rahmen mit Multicast-Zieladresse gelöst. Das spezifizierte Sicherungsprotokoll ist bewußt einfach gehalten und verhindert nicht, daß Pufferengpässe beim Quittungsempfänger auftreten können. Abhilfe schaffen hier Protokolle wie die von [63] oder [111] beschriebenen. Noch genauer ist der Start durch einen SYNC-Rahmen, wofür an den existierenden Monitorkomponenten nur minimale Änderungen und eine Erweiterung des Abfragebusses um dieses Startsignal erforderlich wären.

Während des nunmehr einjährigen Betriebs unterschiedlicher Implementierungsvarianten hat das Meßsystem zahlreiche Erweiterungen erfahren. Dies umfaßt die Verbreiterung des ursprünglich 16 bit breiten Informationskanals auf 18 bit. Die zusätzlichen zwei Bits dienen als Erweiterung der Meßfühlerkennung, können jedoch auch als Erweiterung des Informationsfeldes eines Ereigniseintrags gewertet werden. Die meisten Änderungen erfolgten im Statistikinterpreter sowie an der Schnittstelle zu weiterverarbeitenden Werkzeugen. Gerade durch diese Schnittstelle ist der Wechsel zu Werkzeugen sehr einfach, welche die Ergebnisse in grafischer Form aufbereiten können.

Bei den in Kap. 7 beispielhaft vorgestellten Messungen liefen alle Systeme in realer Umgebung, jedoch waren keine weiteren Aktivitäten auf den Objektsystemen gestartet worden. Reproduzierbare Ergebnisse mit zusätzlicher Systemlast verlangen, daß die auf den einzelnen Objektsystemen ablaufenden Programme kontrolliert angestoßen und während ihrer Laufzeit überwacht werden können. Zu diesem Zweck wurde ein verteilter Lastgenerator [12] entwickelt, der wie das verteilte Meßsystem einen zentralen Steuerungsrechner und verteilt angeordnete Objektrechner umfaßt. Ähnlich dem Dienstprogramm *cron* [181] werden alle zukünftig eingeplanten Aktivitäten in einer Tabelle auf der zentralen Station eingetragen. Nach dem Start des Lastgenerators erzeugt dieser auf allen Objektrechnern Kontrollprozesse, die zum Startzeitpunkt von der zentralen Station eine Befehlszeile inklusive aller Parameter erhalten und den Programmendzustand an die steuernde Station melden. Unter Einsatz des verteilten Lastgenerators wurde der Einfluß lokaler Belastungen auf die Kommunikationsleistungen untersucht. Bild 8.1 zeigt den gemessenen Netzverkehr bei gleichzeitigem Start von n Transportverbindungen am leeren Netz mit real stattfindenden Plattenzugriffen und ohne Plattenzugriffe.

Als Ergebnis der Messungen mit dem Meßsystem sind empirisch gemessene Klassenverteilungen verfügbar. Die weitere Benutzung der gemessenen Werte in Simulation oder Analyse verlangt jedoch oft, daß für diese Werte eine Approximation durch eine bekannte Verteilungsfunktion durchgeführt wird. Die Parameter der Verteilung lassen sich bestimmen, indem die Momente der approximierten an die Momente der empirischen Verteilung angepaßt werden [z.B. in 35, 178]. Als approximierte Verteilungen der oft gemessenen bi- oder trimodalen Verteilungen eignen sich hier:



a) mit Plattenzugriff b) ohne Plattenzugriff

Bild 8.1: Prozentuale Netzlast beim Einsatz des verteilten Lastgenerators (erster Anstieg von ca. 1% infolge der Startaufträge an die Kontrollprozesse)

- hyperexponentielle Verteilung [43, 156],
- Erlang-Mischverteilung [178] oder
- konstante gefolgt von hyperexponentieller Verteilung [35].

Zur Prüfung der Übereinstimmung zwischen der gemessenen und einer angepaßten Verteilung werden der Chi^2-Test und der Kolmogorov-Smirnov-Test [4] durchgeführt.

Manche Forderungen von Anwendern konnten nicht realisiert werden, da die Pflege des Systems viel Einsatz verlangt hat. So bleibt die statistische Auswertung auf das Erkennen von Mustern, Paaren und Serien beschränkt. Einschränkungen infolge der einfachen Zuordnung zu Musterpaaren machen die statistische Auswertung bei asynchron ablaufenden Prozessen in vielen Fällen unmöglich. Auch die Annahme eines Erneuerungsprozesses bei der Statistikauswertung ist nicht immer zulässig.

Eine Auswertung verlangt Einblicke in die Systemabläufe und kann nur durch Interpretation der Gantt-Diagramme oder eine sehr aufwendige Kodierung der Ereignisse erfolgen. So bleiben heute noch viele zeitaufwendige Arbeiten dem Benutzer überlassen.

A Notation der Datenbeschreibungssprache DDL

```
definition_list:              '%' sample_definition_list '%' '%' hypersample_definition_list ;
sample_definition_list:       sample_definition | sample_definition_list sample_definition ;
sample_definition:            sample_nr '{' s_define_list '}' ;
s_define_list:                s_field_definition ';' | s_define_list s_field_definition ';' ;
s_field_definition:           field_name '[' field_length ']' sep field_descript ;
sample_nr:                    dec_num ;
field_name:                   str ;
field_length:                 dec_num ;
field_descript:               'var' | 'fifo' '=' number | 'fifo' ;
number:                       dec_num | hex_num | bin_num ;
hypersample_definition_list:  hypersample_definition
                              | hypersample_definition_list hypersample_definition ;
hypersample_definition:       "" h_sample_name "" sample_nr_list '{' h_define_list '}' ;
h_sample_name:                str ;
sample_nr_list:               sample_nr | sample_nr_list '+' sample_nr ;
h_define_list:                fifo_definition h_list ;
fifo_definition:              <empty> | 'fifo' '=' num ';' ;
h_list:                       <empty> | norm_h_list ;
norm_h_list:                  h_definition_field ';' | norm_h_list h_definition_field ';' ;
h_definition_field:           field_name '=' number ;
str:                          'a..zA..Z' | str 'a..zA..Z_0..9' ;
dec_num:                      '0..9' | dec_num '0..9' ;
hex_num:                      '0' 'h' hex | '0' 'H' hex | '0' 'x' hex | '0' 'X' hex ;
hex:                          'a..fA..F0..9' | hex 'a..fA..F0..9' ;
bin_num:                      '0' 'b' bin | '0' 'B' bin ;
bin:                          '0..1' | bin '0..1' ;
sep:                          ':' | ',' ;
```

Literaturverzeichnis

[1] M.ABRAMS: Design of a Measurement Instrument for Distributed Systems, IBM Research Division, Zurich Research Laboratory, Research Report RZ1639, 1987

[2] H.AKIMA: A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedure, *Journal of ACM*, Vol.17, No.4, October 1970, pp.589-602

[3] ADVANCED MICRO DEVICES: Am 7990 - Local Area Network Controller for Ethernet (LANCE), Datenblatt 1986

[4] J.BANKS, J.S.CARSON: Discrete-Event Simulation, Prentice-Hall Inc., 1984

[5] B.BECK: Anpassung eines Meßadapters an den IBM-AT Bus, Studienarbeit Nr. 990, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990

[6] T.E.BELL, B.W.BOEHM, R.A.WATSON: Framework and Initial Phases for Computer Performance Improvement, AFIPS, No.41, pp.1141-1154

[7] T.BEMMERL, R.LINDHOF, T.TREML: Ein Monitorsystem zur verzögerungsfreien Überwachung von Multiprozessoren, *5. GI/ITG-Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Braunschweig, September 1989, S.51-59

[8] C.J.BENNETT, A.J.HINCHLEY: Measurements of the Transmission Control Protocol, in *Computer Network Protocols*, ed. by A.Danthine, 1978, pp.G1.1- G1.11

[9] R.BERRY, J.HELLERSTEIN: An Approach to Detecting Changes in the Factors Affecting the Performance of Computer Systems, *Performance Evaluation Review*, vol,19, no.1, 1991, pp.39-49

[10] R.BODNARCHUK, R.B.BUNT: A Synthetic Workload Model for a Distributed System File Server, *Performance Evaluation Review*, vol.19, no.1, May 1991, pp.50-59

[11] D.A.BORMAN: Implementing TCP/IP on a Cray computer, *Computer Communication Review*, vol.19, no.2, April 1989, pp.11-15

[12] M.BOSLER: Entwicklung und Implementierung eines verteilten und portablen Lastgenerators (VLG), Studienarbeit Nr. 999, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990

[13] W.BUX, U.HERZOG: The Pase-Concept: Approximation of Measured Data and Performance Analysis, *Proc. Int. Symposium on Computer Performance Modeling , Measurement and Evaluation,* Yorktown Heights, NY, August 1977, pp.23-38

[14] P.CALINGAERT: System Performance Evaluation: A Survey and Appraisal, *Communications of the ACM*, vol.10. no.1, 1967, pp12-18

[15] H.N.CANTRELL, A.L.ELLISION: Multiprogramming System Performance Measurement and Analysis, Proc. AFIPS, SJCC, vol.23, pp.71-80

[16] J.B.CARTER, W.ZWAENEPOEL: Optimistic Implementation of Bulk Data Transfer protocols, Performance Evaluation Review, vol.17, no.1, May 1989, pp.61-69

[17] J.D.CASE, M.FEDOR, M.L.SCHOFFSTALL, C.DAVIN: Simple Network Management Protocol (SNMP), *1157*, May 1990

[18] L.N.CASSEL, C.PARTRIDGE, J.WASTCOTT: Network Management Architectures and Protocols: Problems and Approaches, *IEEE Journal on Selected Areas in Communications*, vol.7, no.7, September 1989, pp.1104-1114

[19] CCITT: Recommendation I.320 - ISDN protocol reference model, Melbourne, 1988, Fascicle III.8, pp.25-34

[20] CCITT: Recommendation I.350 - General aspects of quality of service and networking performance in digital networks, including ISDN, Melbourne, 1988, Fascicle III.8, pp.129-139

[21] CCITT: Recommendation I.352 - network performance objectives for connection processing delays in an ISDN, Melbourne, 1988, Fascicle III.8, pp.140-152

[22] CCITT: Recommendation X.134 - Portion boundaries and packet layer reference events: basis for defining packet-switched performance parameters, Melbourne 1988, Fascicle VIII.3, pp.358-368

[23] CCITT: Recommendation X.135 - Speed of service (delay and throughput) performance values for public data networks when providing international packet-switched services, Melbourne 1988, Fascicle VIII.3, pp.368-391
Supplement No.1 - Some test results from specific national and international portions, Melbourne 1988, Fascicle VIII.3, pp.471-473

[24] CCITT: Recommendation X.136 - Accuracy and dependability performance values for public data networks when providing international packet-switched services, Melbourne 1988, Fascicle VIII.3, pp.392-410

[25] CCITT: Recommendation X.137 - Availability performance values for public data networks when providing international packet-switched services, Melbourne 1988, Fascicle VIII.3, pp.410-421

[26] V.CERF, E.CAIN: The DOD Internet architecture model,*Computer Networks and ISDN Systems*, 10 (1983) 307-318

[27] S.T.CHANSON, M-J.GOH: Implementation of the ISO File Transfer, Access and Management Protocol in the UNIX Environment, *Proc. Computer Networking Symposium*, IEEE, Washington, D.C., April 11-13, 1988, pp.44-52

[28] D.R.CHERITON, C.L.WILLIAMSON: Network Measurement of the VMTP Request-Response Protocol in the V Distributed System, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modelling of Computer Systems*, Banff, Canada, May 11-14, 1987, pp.216-225

[29] D.R.CHERITON, C.L.WILLIAMSON: VMTP as the Transport Layer for High-Performance Distrubuted Systems, *IEEE Communications Magazine*, June 1989, pp.37-44

[30] G.CHESSON, L.GREEN: XTP-Protocol Engine VLSI for Real-Time LANs, *EFOC/LAN'88*, Amsterdam, July 1, 1988

[31] G.CHESSON: XTP/PE Design Considerations, *IFIP WG6.1/WG6.4 Int. Workshop on Protocols for High-Speed Networks*, ed. by H.Rudin + R.Williamson, Zurich, Switzerland, May 9-11, 1989, Elsevier Science Publishers B.V. (North-Holland), pp.27-33

[32] D.D.CLARK, V.JACOBSON, J.ROMKEY, H.SALWEN: An Analysis of TCP Processing Overhead, *IEEE Communications Magazine*, June 1989, pp.23-29

[33] P.B.DANZIG: Finite Buffers and Fast Multicast, *Performance Evaluation Review*, vol.17, no.1, May 1989, pp.108-117

[34] P.B.DANZIG, S.MELVIN: High Resolution Timing with Low Resolution Clocks and A Microsecond Resolution Timer for Sun Workstations, *Operating System Review*, vol.24, no.1, Jan. 1990, pp.23-26

[35] P.B.DANZIG: An Analytical Model of Operating System Protocol Processing Including Effects of Multiprogramming, *Performance Evaluation Review*, vol.19, no.1, may 1991, pp.11-20

[36] R.T.DIMPSEY, R.K.IYER: Performance Prediction and Tuning on a Multiprocessor, *Computer Architecture News*, vol.19, no.3, May 1991, pp.190-199

[37] DEUTSCHES INSTITUT FÜR NORMUNG: Messung und Bewertung der Leistung von DV-Systemen, Entwurf 66273, Teil 1, März 1990

[38] DIGITAL EQUIPMENT CORPORATION: DECnet Digital Network Architecture (Phase IV) - General Description, Bedford, Digital Equipment Corporation, No. AA-149A-TC, 1982,

DIGITAL EQUIPMENT CORPORATION: DECnet Digital Network Architecture (Phase V) - General Description, Bedford, Digital Equipment Corporation, No. EK-DNAPV-GD, September 1987

[39] DIGITAL EQUIPMENT CORPORATION: DECnet Distributed Time Synchronization Service, Bedford, Digital Equipment Corporation

[40] DUDEN - Informatik, Dudenverlag, 1988

[41] U.ELZUR: Getting the most out of Ethernet's data link and transport layers, *Data Communications*, March 21, 1989, pp.53-59

[42] O.ENDRISS, M.STEINBRUNN, M.ZITTERBART: NETMON-II a monitoring tool for distributed and multiprocessor systems, *Performance Evaluation* 12 (1991) 191-202

[43] S.O.FALAKI, S.-A.SORENSEN: Traffic measurements on a local area computer network, *Computer Communications*, vol.15, no.3, April 1992, pp.192-197

[44] G.FÄRBER: Workstations in den 90ern, *Praxis der Kommunikationsverarbeitung und Kommunikation*, PIK 14 (1991) 4, Oktober-Dezember 1991, S.216-222

[45] F.FEHLAU, TH.SIMON, O.SPANIOL, J.SUPPAN-BOROWKA: Messungen des Leistungsverhaltens Lokaler Netze mit einem Software-Monitor, *Informatik Forschung und Entwicklung* (1987) 2: 55-64

[46] D.FERRARI: Computer Systems Performance Evaluation, Prentice Hall, 1978

[47] D.FERRARI, G.SERAZZI, A.ZEIGNER: Measurements and Tuning of Computer Systems, Prentice Hall, 1983

[48] D.FERRARI: On the Foundations of Artificial Workload Design, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modelling of Computer Systems*, Camdridge, MA, August 21-24, 1984, pp.8-14

[49] D.FERRARI: Workload Characterization for Tightly-Coupled and Loosely-Coupled Systems, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modelling of Computer Systems*, Berkeley,CA, May 23-26, 1989, p.210

[50] H.FINK: Aufbau und Test einer intelligenten LAN-Sensorkarte, Studienarbeit Nr. 1074, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1991

[51] P.J.FLEMING, J.J.WALLACE: How not to lie with Statistics: the correct way to summarize benchmark results, *Communications of the ACM*, March 1986, vol.29, no.3, pp.218-221

[52] W.FÖCKELER, N.RÜSING: Aktuelle Probleme und Lösungen zur Leistungsanalyse von modernen Rechensystemen mit Hardware-Meßwerkzeugen, *5. GI/ITG-Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Braunschweig, September 1989, S.39-50

[53] M.FRY: The Efficiency of OSI Protocol Stacks: A Computer Systems Perspective, *ICCC90*, New Delhi, India, November 1990, pp.553-557

[54] S.FUNK: Entwurf komplexer Meßfühler für lokale Netze, Studienarbeit Nr. 901, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1988

[55] S.FUNK: Statistische Auswertung von Meßdaten basierend auf einer universellen Datenbeschreibungssprache, Studienarbeit Nr. 1045, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1991

[56] B.GARBEE: The KA9Q Internet Software Package, User Manual, May 1989

[57] T.GAUWEILER: UNIX als Basis für verteilte Systeme, *Informationstechnik it*; 29. Jahrgang, Heft 6/1987, S.386-390

[58] GFS-Metra: AT/RTX Device Driver for CP536*, Software Design Document, Revision 1.0, GFS-Metra, 21. April 1987

[59] O.GIHR: Meßgrössen und Meßvorschriften für Verkehrsprofile in Lokalen Netzen, *Technischer Anhang zur Projektvereinbarung „Verkehrsmessung bei LAN"*, VDMA und Universität Stuttgart, Stuttgart, 1986

[60] O.GIHR, M.WEIXLER: Messung der Datenverkehrsprofile in Lokalen Netzen, *ITG/GI-Fachtagung „Kommunikation in verteilten Systemen"*, Stuttgart, Februar 1989, pp.861-877

[61] O.GIHR, M.WEIXLER: Messung der Datenverkehrsprofile in Lokalen Netzen, *VDMA-Bericht, 6. Juni 1989*

[62] M.GÖTZER: Aufbau mehrerer Adapterplatinen für WS20, Studienarbeit Nr. 932, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1989

[63] K.J.GOLDMAN: Highly concurrent logically synchronous multicast, *Distributed Computing* (1991) 4:189-207

[64] W.GRAETSCH, H.KÄSTNER: Betriebssystemmessungen mit einem Firmware-Monitor, *2. GI/NTG Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Stuttgart, Februar 1983, S.63-77

[65] P.GUNNINGBERG, M.BJORKMAN, E.NORDMARK, S.PINK, P.SJODIN, J.E.STROMQUIST: Application Protocols and Performance Benchmarks, *IEEE Communications Magazine*, vol.27, no.6, June 1989, pp.30-36

[66] R.GUSELLA, S.ZATTI: The Accuracy of the Clock Synchronisation achieved by TEMPO in Berkeley UNIX 4.3BSD, *IEEE Transactions on Software Engineering*, vol.15, no.7, July 1989, pp.847-853

[67] R.GUSELLA: A Measurement Study of Diskless Workstation Traffic on an Ethernet, *IEEE Transactions on Communications*, vol.38, no.9, September 1990, pp.1557-1568

[68] D.HABAN, D.WYBRANIETZ: A Hybrid Monitor for Behaviour and Performance Analysis of Distributed Systems, *IEEE Transactions on Software Engineering*, vol.16, no. 2, February 1990, pp.197-211

[69] B.HAWE: Technology Implications in LAN Workload Characterization, in *Workload Characterization of Computer Systems and Computer Networks*, ed. by G.Serazzi, Elsevier Science Publishers B.V.(North-Holland), 1986, pp.111-130

[70] S.HEATLEY, D.STOKESBERRY: Measurement of a Transport Implementation Running Over an IEEE 802.3 Local Area Network, *Proc. IEEE Computer Networking Symposium*, IEEE, Washington, DC, April 11-13, 1988, pp.34-43

[71] D.HEGER, K.WATSON: Modelling of Load Patterns and Benchmarks for Performance Evaluation of Local Area Networks, *2.GI/ITG Fachtagung „Messung, Modelliern g und Bewertung von Rechensystemen"*, Stuttgart, Februar1983, S.93-106

[72] D.HEHMANN, B.KÖHLER, N.LUTTENBERGER, L.MACKERT, W.SCHULZ, H.STÜTTGEN: Implementation Experience with a Communication Subsystem Prototype for B-ISDN, *3.rd IFIP WG 6.4 Conference on High Speed Networking*, Berlin, March 18-21, 1991, pp.305-319

[73] J.HELLERSTEIN: A Statistical Approach to Diagnosing Intermittent Performance-Problems Using Monotone Relationships, *Performance Evaluation Review*, vol.17, no.1, May 1989, pp.20-28

[74] O. HENN: MAP/TOP 3.0 - Strategisches Potential und Wirtschaftlichkeit, *CIM Management* 4(1988), S.44-47

[75] H.HERBERTH, W.GORA: Zeitsynchronisation in lokalen Netzen, *GI/NTG Fachtagung "Kommunikation in verteilten Systemen"*, Aachen, Februar 1987, S.453-464

[76] U.HERCKSEN, R.KLAR, W.KLEINÖDER, F.KNEISSL: Measuring Simultaneous Events in a Multiprocessor System, *Performance Evaluation Review*, vol.11, no.4, 1982, pp.77-88

[77] H.HERZOG, F.HILDEBRANDT, H.-O.LEILICH, P.MERTINATSCH, G.STIEGE, H.CH.ZEIDLER: Die Braunschweiger relationale Datenbankmaschine RDBM - Projektergebnisse, *Informatik Forschung und Entwicklung* (1990)5:115-126

[78] HEWLETT-PACKARD COMPANY: HP 4972A LAN protocol analyzer, Hewlett-Packard Company, 1988

[79] HEWLETT-PACKARD COMPANY: HP 4974S MAP protocol analyzer, Hewlett-Packard Company, 1988

[80] HEWLETT-PACKARD COMPANY: HP 5071A Primär-Frequenznormal, Hewlett-Packard Company, 1992

[81] C.A.R.HOARE: Monitors: an operating system structuring concept, *Communications of the ACM*, vol.18, no.8, October 1974, pp.453-457

[82] R.HOFMANN, R.KLAR, N.LUTTENBERGER, B.MOHR: Zählmonitor 4: Ein Monitorsystem für das Hardware- und Hybrid-Monitoring von Multiprozessor- und

Multicomputer- Systemen, *4. GI/ITG-Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Erlangen Sep./Okt. 1987, S.79-99

[83] I.HU: Measuring File Access Patterns in UNIX, *Performance Evaluation Review*, vol.14, no.2, August 1986, pp.15-20

[84] INFORMIX INTERNATIONAL: WINGZ Graphical Spreadsheet, Version 1.0, Informix International Inc., 1989

[85] INTEGRATED DEVICE TECHNOLOGY: IDT7200S/L CMOS parallel First-In/First-Out - 256*9bit, Integrated Device Technology Inc., Datenblatt 1987

[86] IEEE: Multibus/IEEE-796 Bus Specification, IEEE, 1982

[87] IEEE: Personal Computer Bus Standard P996, Draft D2.00, IEEE, January 1990

[88] INTEL CORPORTION: iNA 960 Programmer´s Reference Manual, Intel Corporation, 1984

[89] INTEL CORPORTION: iNA 960 Architecture Reference Manual, Intel Corporation, 1984

[90] INTEL CORPORTION: Microcommunications Handbook, Intel Corporation, 1987

[91] INTEL CORPORATION: iAPX286 Programmer's Reference Manual, Intel Corporation, 1985
INTEL CORPORATION: iAPX286 Operating Systems Writer's Guide, Intel Corporation, 1983

[92] INTEL CORPORATION: i82586 - Local Area Network Coprocessor, Intel Corporation, Datenblatt 1988

[93] INTEL CORPORATION: iRMX 86 Operating System, Intel Corporation, 1985

[94] INTEL CORPORATION: Extended iRMX II Operating System, Intel Corporation, 1988

[95] INTEL CORPORATION: System 310AP, Intel Corporation

[96] ISO 7498: Information Processing Systems - Open Systems Interconnection - Basic Reference Model, International Standard IS 7498, October 1984
Part 4: Management Framework, 1989

[97] ISO 8073: Information Processing Systems - Open Systems Interconnection - Connection Oriented Transport Protocol Specification, 1984

[98] ISO 8802/2: Information Processing Systems - Open Systems Interconnection - Logical Link Control, 1986
Annex A: Acknowledged Connectionless Service, October 1986

[98] ISO 8802/3: Information Processing Systems - Open Systems Interconnection - Carrier Sense/Multiple Access with Collision Detection, 1984

[100] ISO 9595: Information Processing Systems - Open Systems Interconnection - Common Management Information Services Definition (CMIS), 1990

[101] ISO 9596: Information Processing Systems - Open Systems Interconnection - Common Management Information Protocol Specification (CMIP), 1990

[102] ISO 10040: Information Processing Systems - Open Systems Interconnection - Systemy Management Overview, 1990

[103] ISO 10164-4: Information Processing Systems - Open Systems Interconnection - Management Information Services - Systemy Management - Part 4: Alarm Reporting Function, June 1991

[104] ISO 10164-5: Information Processing Systems - Open Systems Interconnection - Management Information Services - Systemy Management - Part 5: Event Reporting Function, June 1991

[105] ISO 10164-6: Information Processing Systems - Open Systems Interconnection - Management Information Services - Systemy Management - Part 6: Log Control Function, June 1991

[106] ISO 10164-11: Information Processing Systems - Open Systems Interconnection - Management Information Services - Systemy Management - Part 11: Worklad Monitoring Function, October 1991

[107] ISO 10165-2: Information Processing Systems - Open Systems Interconnection - Structure of Management Information- Part 2: Definition of Management Information, June 1990

[108] R.JAIN, I.CHLAMTAC: The P^2 Algorithm for Dynamic Calculation of Quantiles and Histograms Without Storing Observations, *Communications of the ACM*, vol.28, no.10, October 1985, pp.1076-1085

[109] J.B.JOHNSON, S.KASSEL: The Multibus Design Guidebook - Structures, Architectures, and Applications, McGraw-Hill Book Company 1984

[110] M.JOHNSON: Network Protocol Performance, *Performance Evaluation Review*, vol.17, no.1, May 1989, p.217

[111] M.F.KAASHOEK, A.S.TANENBAUM, S.HUMMEL, H.E.BAL: An efficient reliable broadcast protocol,*Operating System Review*, vol.23, October 1989, pp.5-19

[112] M.KATZ, G.BIWER, K.BENDER: Die PROFIBUS-Anwendungsschicht, *Automatisierungstechnische Praxis* atp 31(1989)12, S.588-597

[113] T.KEROLA, H.SCHWETMAN: Monit: A Performance Monitoring Tool for Parallel and Pseudo-Parallel Programs, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modelling of Computer Systems*, Banff, Canada, May 11-14, 1987, pp.163-174

[114] R.KLAR: Hardware-/Software-Monitoring, *Informatik-Spektrum*, (1985) 8:37-38

[115] R.KLAR: Integrating Monitoring and Modeling for Evaluation of Multiprocessor Systems, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modelling of Computer Systems*, Berkeley, CA, May 23-236 1989, p.221

[116] D.E.KNUTH: The Art of Computer Programming - Volume 2, Addison-Wesley Publishing Company, 1969

[117] H.-E.KÖCHLING: Zusammenfassung zeitlich verschobener Ereignisspuren, Studienarbeit Nr. 1128, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992

[118] H.KOPETZ, W.OCHSENREITER: Clock Synchronization in Distributed Real-Time Systems, *IEEE Transactions on Computers*, vol.36, no.8, August 1987, pp.933-940

[119] R.KOXHOLT: Die Rolle von ISDN in der Bürokommunikation, *DATACOM*, 7/88, S.80-88

[120] K.KÜMMERLE: Höhere Programmiersprachen und ihre Compiler, Vorlesungsskript, Universität Stuttgart

[121] L.LAMPORT: Time, Clocks, and the Ordering of Events in a Distributed System, *Communications of the ACM*, vol.21, no.7, July 1978, pp.558-565

[122] P.L'ECUYER: Efficient and Portable Combined Random Number Generators, *Communications of the ACM*, vol.31, no.6, June 1888, pp.742-774

[123] T.P.LEE, R.E.BARKELEY: Configuring UNIX STREAMS Communication Buffers Based on an Erlang Traffic Model, *Proc. IEEE Computer Networking Symposium*, Washington, DC, April 11-13, 1988, pp.230-234

[124] F.LEPAGE, K.DOUADI, J.-Y.BRON: Temporal performance measurements for MAP network, *2nd. Int. Conference on Local Communcation Systems: LAN and PBX*, Palma de Mallorca, Spain, June 26-28, 1991, pp.179-192

[125] H.M.LEVY, D.W.W.CLARK: On the Use of Benchmarks for Measuring System Performance, *Computer Architecture News*, vol.10, no.6, December 1982

[126] W.P.LIDINSKY: Data Communications Needs, *IEEE Network Magazin*, March 1990, pp.28-33

[127] R.P.LIPPMANN: An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, April 1987, pp.4-22

[128] E.MAFLA, B.BHARGAVA: Communication Facilities for Distributed Transaction-Processing Systems, *IEEE Computer*, August 1991, pp.61-66

[129] MAP: MAP Network Architecture, *MAP 3.0 Implementation Release, General Motors Technical Center*, April 1987

[130] MAP/TOP 3.0: Network Management Requirements Specification, Draft, *General Motors Technical Center*, March 1987

[131] M.V.MARATHE, R.A.SMITH: Performance of a MAP Network Adapter, *IEEE Network*, May 1988, vol.2, no.3, pp.82-88

[132] S.MAZUMDAR, A.A:LAZAR: Monitoring Integrated Networks for Performance Management, *ICC'90*, pp.302.1.1-302.1.6

[133] K.MCCLOGHRIE, M.T.ROSE: Management Information Base for network management of TCP/IP-based internets, *RFC 1156*, May 1990

[134] G.MCDANIEL: The Mesa Spy: An Interactive Tool for Performance Debugging, *Performance Evaluation Review*, vol.11, no.4, 1982, pp.68-76

[135] P.MCKERROW: Performance Measurement of Computer Systems, Addison-Wesley Publishing Company, 1987

[136] H.MEISNER: High Performance SCP - Performance Analyse, SIEMENS, Bericht SINEC88-009/001, ESTE 36, 18.4.1988

[137] S.W.MELVIN, Y.N.PATT: The Use of Microcode Instrumentation for Development, Debugging and Tuning of Operating System Kernels, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modeling of Computer Systems*, Santa Fe, New Mexico, May 24-27, 1988, pp.207-214

[138] B.MILLER: DPM: A Measurement System for Distributed Programs, *IEEE Transactions on Computers*, vol.37, no.2, February 1988, pp.243-248

[139] D.L.MILLS: Measured performance of the Network Time Protocol in the Internet system, *RFC 1128,* October 1989

[140] D.L.MILLS: Internet time synchronization: the Network Time Protocol, *RFC 1129,* October 1989

[141] D.L.MILLS: On the Accuracy and Stability of Clocks Synchronized by the Network Time Protocol in the Internet System, *Computer Communication Review (SIGCOMM)*, vol.20, no.1, January 1990, pp.65-75

[142] A.MINK, J.DRAPER, J.ROBERTS, R.CARPENTER: Hardware-Assisted Multiprocessor Performance Measurement, in *PERFORMANCE'87*, ed. by P.-J.Courtois, G.Latouche, Elsevier Science Publishers B.V.(North-Holland), 1988, pp.151-168

[143] A.MINK, R.CARPENTER, G.NACHT, J.ROBERTS: Multiprocessor Performance-Measurement Instrumentation, *IEEE Computer*, September 1990, pp.63-75

[144] D.MIRCHANDANI, P.BISWAS: Charatcerization and Modeling Ethernet Performance of Distributed DECwindows Applications, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modelling of Computer Systems*, Boulder, CO, May 22-25, 1990, pp.253-254

[145] J.C.MOGUL, R.F.RASHID, M.J.ACCETTA: The Packet Filter: An Efficient Mechanism for User-level Network Code, *Operating Systems Review (SIGOPS)*, vol.21, no.5, 1987, pp.39-51

[146] B.MORTAZAVI: Performance of MAP in the Remote Operation of a CNC, *IEEE Transactions on Software Engineering*, vol.16, no. 2, February 1990, pp.231-237

[147] MOTOROLA: MC68824 Token Bus Controller, Motorola Inc., Datenblatt 1986

[148] W.MÜNCH: Unterstützen TCP/IP- und ISO/OSI-Protokolle, *Elektronik*, 23/1991, S.200-206

[149] NATIONAL SEMICONDUCTOR: DP 83932 SONIC Systems-oriented Network Interface Controller, National Semiconductor, Datenblatt, Mai 1990

[150] D.NEIBAUR: Understanding XNS: The prototypical internetwork protocol, *Data Communications*, September 1989, pp.43-51

[151] NETWORK GENERAL: The Sniffer, Network General Corporation, 1988

[152] G.NUTT: Tutorial: Computer System Monitors, *Computer*, vol. 8, no.11, November 1975, pp.51-61

[153] T.OEXNER: Spezifikation und Implementierung eines sicheren Protokolles auf Basis von LLC für UNIX-Rechner, Studienarbeit Nr. 977, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990

[154] A.PARK, J.C.BECKER: Measurements of the Paging Behaviour of UNIX, *Performance Evaluation Review*, vol.19, no.1, may 1991, pp.216-217

[155] D.PARTRIDGE, R.CARD: Hardware Monitoring of Real-Time Aerospace Computer Systems, *Proc. of Int. Symp. on „Computer Performance Modelling, Measurement and Evaluation"*, Harvard Univ., Cambridge, MA, March 29-31, 1976, pp.85-101

[156] G.PAVLOU, G.KNIGHT: Basic rate ISDN workstation traffic patterns, *Computer Communications*, vol.13, no.10, December 1990, pp.587-594

[157] V.PAXSON: Measurements and Models of Wide Area TCP Conversations, *Lawrence Berkeley Laboratory*, Report LBL-30840, Berkeley, CA, May 1991

[158] G.-S.POO: Transport and link layer performance comparison of CSMA/CD, token bus and token ring networks, *Computer Communications*, vol.14, no.4, May 1991, pp.235-243

[159] J.B.POSTEL: Internet Protocol, *RFC 791*, September 1981

[160] J.POSTEL, J.REYNOLDS: A Standard for the Transmission of IP Datagrams over IEEE 802 Networks, *RFC 1042*, February 1988

[161] A.POTTHAST, R.-N.RENNERT: Vernetzte, automatisierte Pressen, *Elektronik*, Nr. 3, 2.2.1990, S.70-76

[162] DIN V 19245: Teil 1: PROFIBUS - Vornorm, Beuth-Verlag, Berlin

[163] PROTOCOL ENGINES: XTP Protocol Definition, Revision 3.4, 17 July 1989, Protocol Engine, Inc, 1421 State Street, Santa Barbara, CA 93101

[164] C.PU, F.KORZ, R.C.LEHMAN: An Experiment on Measuring Application Performance over the Internet, *Performance Evaluation Review*, vol.19, no.1, May 1991, pp.220-221

[165] J.S.QUARTERMAN, A.SILBERSCHATZ, J.L.PETERSON: 4.2BSD and 4.3BSD as Examples of the UNIX Sytem, *ACM Computing Survey*, vol.17, no.4, December 1985, pp.379-418

[166] A.QUICK: Synchronisierte Software-Messung zur Bewertung des dynamischen Verhaltens eines UNIX-Multiprozessor-Betriebssystems, *5. GI/ITG-Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Braunschweig, September 1989, S.142-158

[167] R.R.RAZOUK, T.STEWART, M.WILSON: Measuring Operating System Performance on Modern Micro-Processors, *Performace'86 and SIGMETRICS Joint Conference on Computer Performance Modelling, Measurements and Evaluation*, Raleigh, NC, May 27-30, 1986, pp.193-202

[168] D.RETZ: TCP/IP: DOD suite marches into the business world, *Data Communications*, November 1987, pp.209-225

[169] J.RIEMER: Entwurf eines multifunktionalen LAN-Sensors, Studienarbeit Nr. 1040, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990

[170] G.RÖSSLER, W.SCHOLLENBERGER: Netzmanagement in heterogenen lokalen Netzen, *Praxis der Kommunikationsverarbeitung und Kommunikation*, PIK 14 (1991) 4, Oktober-Dezember 1991, S.211-215

[171] TH.J.ROUTT: From TOP (3.0) to bottom: Architectural close-up, *Data Communications*, May 1988, pp.237-256

[172] RTCS: AT/RTX Real-Time Multi-Tasking Operating System, Version 1.3, RTCS/Real-Time Computer Science Corporation, CA

[173] RTCS: RTX286 Real-Time Multi-Tasking Operating System, Installation and Users Guide, Version 2.0, RTCS/Real-Time Computer Science Corporation, CA, October 1986

[174] H.SAAL: LAN Downtime: Clear and Present Danger, *Data Communications*, March 21, 1990, pp.67-72

[175] R.SANDBERG: The SUN Network File System: Design, Implementation and Experience, SUN Microsystems, Inc., 2550 Garcia Ave., Mountain View, CA 94043

[176] C.H.SAUER, K.M.CHANDY: Computer System Performance Modeling, Prentice Hall, 1981

[177] J.SCHIEFER: Portierung und Leistungsuntersuchung einer Multitasking-TCP/IP-Protokollimplementierung für das Betriebssystem XENIX, Studienarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1992

[178] L.SCHMICKLER: Erweiterung des Verfahrens MEDA zur analytischen Beschreibung empirisch gemessener Verteilungsfunktionen, *5. GI/ITG-Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Braunschweig, September 1989, S.175-189

[179] N.SCHMITZ: Statistik - Fehler, Fallen, Schwindel, *6. GI/ITG-Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Neubiberg, September 1991, S.1-14

[180] W.SCHULZ: Entwurf und Aufbau eines Meßadapters für Multibus I, Studienarbeit Nr. 994, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1990

[181] SCO: XENIX System V, Release Notes, Version 2.2.1, Santa Cruz Operation, 1987

[182] SEEQ: 8005 - Advanced Ethernet Data Link Controller, Seeq Technologies, Datenblatt 1988

[183] H.SELEGRAD, K.HOFFMANN, T.NOVACEK: MOP - ein Monitoring-Prozessor für den parallelen Systembus des Multibus II, *Design und Elektronik*, 26. Ausgabe, 22.12.1989, S.80-82

[184] SIEMENS: LAN Protocol-Tester B5100 für CSMA/CD-Netze - Bedienhandbuch, Siemens AG, 1989

[185] SIEMENS: Basis System Host - BSH/UNIX-Pflichtenheft, Version 1.01, Dok.Nr. 352/88/19, Siemens AG, 30. August 1988
GUD: X-IHI-Erweiterung für Ebene 2, GUD Datensysteme GmbH, 19. Juli 1989
GUD: Nachtrag X-IHI, GUD Datensysteme GmbH, 8. August 1989
GUD: Installationshinweise, GUD Datensysteme GmbH, November 1989

[186] SIEMENS: Schnittstellenbeschreibung zur Firmware auf der CP536*, Siemens AG, 27. April 1987

[187] SIEMENS: SINEC File Transfer Service, Beschreibung, Version 1.3, Siemens AG, September 1987
SINEC FTS Version 1.4 für SICOMP PC16-20 unter XENIX V, Siemens AG
SINEC FTS unter iRMX II, Version 1.0, Siemens AG, Oktober 1989

[188] SIEMENS: Rechnersystem SICOMP - Personal Computer SICOMP PC16-20, Systemhandbuch, Siemens AG, März 1987

[189] J.SMITH, G.Q.MAGUIRE: Measured Response Times for Page-Sized Fetches on a Network, *Computer Architecture News*, vol.17, no.5, September 1989

[190] R.SNODGRASS: A Relational Approach to Monitoring Complex Systems, *ACM Transactions on Computer Systems*, vol.6, no. 2, May 1988, pp.157-196

[191] P.R.STRAUSS: Flexibility or heresy? The struggle to redraw MAP, *Data Communications*, November 1988, pp.49-56

[192] W.T.STRAYER, A.C.WEAVER: Performance Measurement of Data Transfer Services in MAP, *IEEE Network*, May 1988, vol.2, no.3, pp.82-88

[193] M.STUMM: Verteilte Systeme: Eine Einführung am Beispiel V, *Informatik-Spektrum* , (1987) 10:246-261

[194] C.B.STUNKEL, W.K.FUCHS: TRAPEDS: Producing Traces for Multicomputers Via Execution Driven Simulation, *Performace Evaluation Review*, vol.17, no.1, May 1989, pp.70-78

[195] C.B.STUNKEL, B.JANSSENS, W.K.FUCHS: Address Tracing for Parallel Maschines, *IEEE Computer*, January 1991, pp.31-38

[196] SUN MICROSYSTEMS: RPC: Remote Procedure Call - Protocol Specification - Version 2, *RFC 1057*, June 1988

[197] L.SVOBODOVA: Computer Performance Measurement and Evaluation Methods: Analysis and Applications, Elsevier Computer Science Library, 1976

[198] J.POSTEL, J.REYNOLDS: Telnet Protocol Specification, *RFC 854*, May 1983
J.POSTEL, J.REYNOLDS: Telnet Option Specification, *RFC 855*, May 1983
J.POSTEL, J.REYNOLDS: Telnet Binary Transmission, *RFC 856*, May 1983
J.POSTEL, J.REYNOLDS: Telnet Echo Option, *RFC 857*, May 1983
J.POSTEL, J.REYNOLDS: Telnet Sppress and Go Ahead Option, *RFC 858*, May 1983
J.POSTEL, J.REYNOLDS: Telnet Status Option, *RFC859*, May 1983
J.POSTEL, J.REYNOLDS: Telnet Timing Mark Option, *RFC 860*, May 1983
J.POSTEL, J.REYNOLDS: Telnet Extended Option List, *RFC 861*, May 1983
J.POSTEL, J.REYNOLDS: Telnet End-Of-Record Option, *RFC 855*, May 1983
D.WAITZMAN: Telnet Window Size Option, *RFC 1073*, October 1988
C.HEDRICK: Telnet Terminal Speed Option, *RFC 1079*, December 1988
C.HEDRICK: Telnet Remote Flow Control Option, *RFC 1080*, November 1988
J.VANBOKKELEN: Telnet Terminal-Type Option, *RFC 1091*, February 1989
D.BORMAN: Telnet Linemode Option, *RFC 1116*, August 1989
R.BRADEN: Requirements for Internet Hosts - Application and Support, *RFC 1123*, October 1989

[199] The Astronomical Almanac, Washington, 1984

[200] J.J.P.TSAI, K-Y.FANG, H-Y.CHEN: A Noninvasive Architecture to Monitor Real-Time Distributed Systems, *IEEE Computer*, March 1990, pp.11-23

[201] N.VANDERLIPP, M.ABRAMS, J.CALLAHAN, A.AGRAWALA: Implementation and Measurement of a Distributed Dining Philosophers Algorithm on ZMOB, *University of Maryland, Computer Science Technical Report Series*, TR-1530, August 1985

[202] W-H.WANG, J-L.BAER: Efficient Trace-Driven Simulation Methods for Cache Performance Analysis, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modelling of Computer Systems*, Boulder, CO, May 22-25, 1990, pp.27-36

[203] R.WEICKER: Benchmarking: Status, Kritik und Aussichten, *6. GI/ITG-Fachtagung „Messung, Modellierung und Bewertung von Rechensystemen"*, Neubiberg, September 1991, S.259-277

[204] P.J.WEINBERGER: Cheap Dynamic Instruction Counting, *Bell Systems Technical Journal*, vol.63, no.8, October 1984, pp.1815-1826

[205] M.WEIXLER: Distributed Measurement System for Protocols and Applications in ISO 8802/3 LANs, *4th. International Conference on Data Communication Systems and their Performance*, Barcelona, Spain, June 20-22, 1990, pp.448-455

[206] M.WEIXLER, M.SIEGEL, H.STOLZ: Serieller 48 bit Mustervergleicher, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, Datenblatt 1991

[207] D.J.WRIGHT, M.TO: Telecommunication Applications of the 1990s and their Transport Requirements, *IEEE Network Magazine*, March 1990, pp.34-40

[208] D.WYBRANIETZ, D.HABAN: Monitoring and Performance Measuring Distributed Systems during Operation, *Proc. ACM SIGMETRICS, Conf. on Measurement and Modeling of Computer Systems*, Santa Fe, New Mexico, May 24-27, 1988, pp.197-206

[209] D.WYBRANIETZ: Multicast-Kommunikation in verteilten Systemen, *Informatik Fachberichte 242*, Springer-Verlag, 1990

[210] XEROX: Internet Transport Protocol, XSIS 028112, December 1981,
XEROX: Courier - The Remote Procedure Call Protocol, XSIS 038112, December 1981,
XEROX: Clearinghouse Protocol, XSIS 078404, April 1984,
XEROX: Clearinghouse Entry Formats, XSIS 168404, April 1984,
XEROX: Time Protocol, XSIS 088404, April 1984,
XEROX: Authentication Protocol, XSIS 098404, April 1984,
XEROX: Printing Protocol, XSIS 118404, April 1984, Xerox Corp., Stanford

[211] C-Q.YANG, B.P.MILLER: Performance Measurement for Parallel and Distributed Programs: A Structured and Automatic Approach, *IEEE Transactions on Software Engineering*, vol.15, no.12, December 1989, pp.1615-1629

[212] F.ZWIRNER: Anpassung der TCP/IP-Protokollsuite auf einen ISO Protokollstack, Studienarbeit Nr. 1117, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1991