53. Bericht über verkehrstheoretische Arbeiten

EIN KONZEPT ZUR LEISTUNGSMESSUNG IN VERTEILTEM RECHENSYSTEMEN MIT DEZENTRALEN ZEITGEBERN

von

Michael Johannes Weixler

1993
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 protocol 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

Inhaltsverzeichnis ................................................................. 1
Abkürzungen ............................................................................ 5
Formelzeichen ........................................................................... 6

1 Einleitung ............................................................................. 8
  1.1 Kommunikationsarchitekturen ........................................... 9
  1.2 Ziel der Leistungsmessung ............................................. 11
  1.3 Übersicht über die Arbeit .............................................. 13

2 Leistungsmessung ............................................................... 16
  2.1 Ansätze zur Leistungsmessung ...................................... 16
    2.1.1 Definitionen ......................................................... 16
    2.1.2 Meßansätze ......................................................... 17
    2.1.3 Meßpunkte .......................................................... 22
    2.1.4 Meßverfahren ...................................................... 27
  2.2 Meßkonzepte ................................................................. 29
    2.2.1 ISO OSI-Leistungsmessung .................................... 30
    2.2.2 Leistungsmessung im Internet ............................... 32
    2.2.3 Messung mit einem Monitor .................................. 33
  2.3 Monitore ......................................................................... 34
    2.3.1 Hardware-Monitor ............................................... 37
    2.3.2 Software-Monitor ................................................ 39
    2.3.3 Hybrid-Monitor ................................................... 42

3 Meßproblematik in verteilten Systemen ............................... 44
  3.1 Leistungsmessung an einem System ............................... 44
  3.2 Leistungsmessung an verteilten Systemen ....................... 44
    3.2.1 Messung an einem Meßobjekt ................................ 44
    3.2.2 Messung an mehreren Meßobjekten ...................... 44
  3.3 Synchronisation ............................................................ 46
  3.4 Speicherung der Meßinformation .................................... 49
  3.5 Auswertung .................................................................... 51
    3.5.1 Fehlersuche ......................................................... 51
4 Architektur für ein verteiltes Meßsystem ............................................. 56
  4.1 Hybridmonitor ................................................................. 56
   4.1.1 Hardware-Meßfühler ...................................................... 60
      4.1.1.1 Netzzugang .......................................................... 61
      4.1.1.2 LAN-Koprozessor .................................................. 61
      4.1.1.3 Bussystem ............................................................ 63
   4.1.2 Software-Meßstatements ................................................ 65
   4.1.3 Externe Monitorkomponente ............................................ 66
      4.1.3.1 Abfragestufe ........................................................ 66
      4.1.3.2 Lokale Systemzeit ................................................ 67
      4.1.3.3 Meßspeicher ........................................................ 68
      4.1.3.4 Steuerung ............................................................ 68
  4.2 Zentrale Meßsteuerung ....................................................... 69
   4.2.1 Master-Slave-Konfiguration ......................................... 69
   4.2.2 Kommunikationsprotokoll ............................................. 70
  4.3 Synchronisation ............................................................... 71
   4.3.1 Lokale Zeiterfassung .................................................. 71
   4.3.2 Synchronisationsunterstützung ..................................... 72
   4.3.3 Resynchronisation ...................................................... 73
  4.4 Meßdatenauswertung .......................................................... 78
   4.4.1 Datenbeschreibungssprache ......................................... 79
   4.4.2 Statistikauswertung .................................................... 82
      4.4.2.1 Statistische Größen .............................................. 82
      4.4.2.2 Meßauswertung .................................................... 83

5 Realisierung der Meßteile ...................................................... 84
  5.1 Meßfühlerentwicklungen .................................................... 86
   5.1.1 Meßfühler am Netzzugang ............................................. 86
      5.1.1.1 Meßfühler am LAN-Koprozessor ................................ 86
      5.1.1.2 Meßfühler direkt am Netzzugang ............................... 91
   5.1.2 Meßfühler am Systembus .............................................. 95
   5.1.3 Integration aller Meßfühler .......................................... 97
  5.2 Meßstatements ................................................................. 98
  5.3 Meßkomponente ............................................................... 99
   5.3.1 Steuerungsbau gruppe ................................................ 99
5.3.2 Meßbaugruppe ................................................................. 99
5.4 Realisierung der Meßsteuerung ........................................... 101
5.4.1 Master-Slave-Betrieb ..................................................... 101
5.4.2 Kommunikationsprotokoll .............................................. 101
5.4.3 Befehlsatz ................................................................. 104
5.5 Zusammenfassung der Ereignisspuren ................................... 105
5.5.1 Korrelation ............................................................... 105
5.5.2 Interpolation ............................................................. 109
5.5.3 Sortierung ............................................................... 110
5.6 Implementierung des Statistikinterpreters ............................... 111
5.6.1 Aufbau der Referenzdatenbank aus den Meßdaten ................. 111
5.6.2 Statistische Interpretation der Daten aus der Referenzdatenbank ........................................................................ 112
5.6.2.1 Statistische Auswertung bezüglich Ereignistypen .............. 112
5.6.2.2 Statistische Auswertung bezüglich Ereignispaaren .......... 113
5.6.2.3 Statistische Auswertung bezüglich gesamter Ereignisspur .......... 117

6 Güte und Leistungsbewertung des Meßsystems ......................... 118
6.1 Erzeugung der Meßinformation ............................................ 118
6.1.1 Erzeugung der Meßinformation durch Hardware-Meßfühler ...... 118
6.1.1.1 Netzzugang ............................................................ 118
6.1.1.2 Systembus ............................................................. 121
6.1.2 Erzeugung der Meßinformation durch Meßstatements .......... 122
6.2 Meßgenauigkeit ............................................................. 123
6.2.1 Uhrengenauigkeit ......................................................... 123
6.2.2 Synchronisation mittels Netzverkehr ................................ 125
6.2.3 Synchronisation mittels synthetischem Verkehr .................. 126
6.3 Rechengenauigkeit ......................................................... 127
6.3.1 Berechnungsgenauigkeit bei der Zeitsynchronisation .......... 127
6.3.2 Berechnungsgenauigkeit im Statistikinterpreter ................. 127
6.4 Meßdatenvolumen ......................................................... 128
6.4.1 Datenvolumen aufgrund einer Messung ............................ 128
6.4.2 Datenvolumen für Statistikauswertung ............................ 129
6.5 Rechenzeiten ............................................................... 129
6.5.1 Meßdauer ................................................................. 129
6.5.1 Transferdauer ........................................................... 130
6.5.2 Korrelationsberechnungen ........................................... 131
6.5.3 Statistikauswertung ..................................................... 132

7 Einsatz des verteilten Meßsystems ........................................... 134
7.1 Messungen an einem Objektsystem ..................................... 134
7.1.1 IHI-Treiber auf WS20 unter RTX286 ............................... 134
7.1.2 CNMA-Gateway auf Intel310 unter iRMX II .................................................. 135
7.1.3 TELNET-Applikation ............................................................................. 139
7.2 Messungen an mehreren Objektsystemen ....................................................... 140
7.2.1 Synchroner Filetransfer ........................................................................ 141
7.2.2 Asynchroner Filetransfer ...................................................................... 141

8 Zusammenfassung und Schlußbemerkungen ................................................... 144

A Notation der Datenbeschreibungssprache DDL ............................................ 147

Literaturverzeichnis ......................................................................................... 149
### Abkürzungen

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

\[ \Delta t_i \] \hspace{1cm} \text{Interner zeitlicher Abstand in einer Musterserie}  
\[ \Delta T \] \hspace{1cm} \text{Zeitlicher Abstand der begrenzenden Muster einer Musterserie oder eines Musterpaares}  
\[ \Delta Z \] \hspace{1cm} \text{Maximale Uhrenabweichung}  
\[ [Z] \] \hspace{1cm} \text{Größte ganze Zahl kleiner oder gleich Z}  
\[ c \] \hspace{1cm} \text{Korrelationskoeffizient}  
\[ d \] \hspace{1cm} \text{Fehlerabweichung}  
\[ f \] \hspace{1cm} \text{Teilerfaktor für Frequenz}  
\[ g \] \hspace{1cm} \text{Globale Meßfühleradresse}  
\[ l \] \hspace{1cm} \text{Lokale Meßfühleradresse}  
\[ s \] \hspace{1cm} \text{Anzahl wartender Ereignisse am Abfragebus}  
\[ A \rightarrow B \] \hspace{1cm} \text{Paarzuordnung: Muster A gefolgt von Muster B}  
\[ D \] \hspace{1cm} \text{Dynamisches Korrelationsintervall}  
\[ F \] \hspace{1cm} \text{Oszillatortfrequenz}  
\[ F_i \] \hspace{1cm} \text{Folge der SYNC-Zeintervalle aus Spur i}  
\[ F_{\text{Ein}} \] \hspace{1cm} \text{Einschreibfrequenz}  
\[ F_{\text{Aus}} \] \hspace{1cm} \text{Auslesefrequenz}
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Meßfahreridentifikation</td>
</tr>
<tr>
<td>K</td>
<td>Anzahl von Korrelationsrechnungen</td>
</tr>
<tr>
<td>K_i</td>
<td>Obere Schanke der Klasse i</td>
</tr>
<tr>
<td>K_Schritt</td>
<td>Klassenschrittbreite</td>
</tr>
<tr>
<td>L</td>
<td>Länge des Messpeichers</td>
</tr>
<tr>
<td>L_reale</td>
<td>Reale FIFO-Speicherdauer</td>
</tr>
<tr>
<td>L_virt</td>
<td>Virtuelle FIFO-Speicherdauer</td>
</tr>
<tr>
<td>M</td>
<td>Messdauer</td>
</tr>
<tr>
<td>M[i,j]</td>
<td>Wert aus Verschiebematrix von Spur i mit Spur j</td>
</tr>
<tr>
<td>R_1</td>
<td>Maximale Messdatenrate je Meßfühler</td>
</tr>
<tr>
<td>R_Z</td>
<td>Resynchronisationsrate i.A. von ΔZ</td>
</tr>
<tr>
<td>S</td>
<td>Statisches Korrelationsintervall</td>
</tr>
<tr>
<td>T</td>
<td>Zählerzyklusdauer</td>
</tr>
<tr>
<td>T_A</td>
<td>Auftragsbearbeitungszeit</td>
</tr>
<tr>
<td>T_B</td>
<td>Geschätzte Bearbeitungszeit</td>
</tr>
<tr>
<td>T_E</td>
<td>Erfassungszeit</td>
</tr>
<tr>
<td>T_Eu</td>
<td>Untere Grenze der Erfassungszeit</td>
</tr>
<tr>
<td>T_E0</td>
<td>Obere Grenze der Erfassungszeit</td>
</tr>
<tr>
<td>T_H</td>
<td>Hinlaufzeit</td>
</tr>
<tr>
<td>T_i</td>
<td>Bearbeitungszeit</td>
</tr>
<tr>
<td>T_Abstand</td>
<td>Minimaler Rahmenabstand</td>
</tr>
<tr>
<td>T_Bus</td>
<td>Verzögerung im Systembus-Meßfühler</td>
</tr>
<tr>
<td>T_D</td>
<td>FIFO-Durchfallzeit</td>
</tr>
<tr>
<td>T_K6</td>
<td>Verzögerungszeit bei Transfererkennung</td>
</tr>
<tr>
<td>T_K7</td>
<td>Verzögerungszeit bei SYNC-Erkennung</td>
</tr>
<tr>
<td>T_M</td>
<td>Abfragezeit je Meßfühler</td>
</tr>
<tr>
<td>T_R</td>
<td>Rücklaufzeit</td>
</tr>
<tr>
<td>T_Rahmen</td>
<td>Minimale Rahmendauer</td>
</tr>
<tr>
<td>T_R_E</td>
<td>Verzögerung im Meßfühler bei SYNC-Erkennung bis Rahmenende</td>
</tr>
<tr>
<td>T_SCB</td>
<td>Zeit bis Zugriff auf Status Control Block (SCB)</td>
</tr>
<tr>
<td>T_SR</td>
<td>Verzögerung im Meßfühler bei Transfererkennung</td>
</tr>
<tr>
<td>T_SYNC</td>
<td>Maximal erlaubter SYNC-Abstand</td>
</tr>
<tr>
<td>T_V</td>
<td>Rahmenverzögerung bei kollisionsbehaftetem Senden</td>
</tr>
<tr>
<td>T_j</td>
<td>j-tes Ereignis in Spur i oder</td>
</tr>
<tr>
<td>j-tes Abstandsintervall zwischen SYNC-Ereignissen in Spur i</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>Länge des Abfragezyklus</td>
</tr>
<tr>
<td>ZS</td>
<td>Zeitstempelinhalt</td>
</tr>
</tbody>
</table>
1 Einleitung


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.


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 Netzerwerke entworfen wurden.
1.1 Kommunikationsarchitekturen


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.


Bild 1.1: Komponenten eines zu beobachtenden Systems für die Messung

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,
  Kanalzugriffsvorverfahren,
  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].


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).
<table>
<thead>
<tr>
<th>Protokolle (Schnittstelle)</th>
<th>Stationsanzahl</th>
<th>Netz</th>
<th>Verkehr</th>
<th>Literatur</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netzwerk-Schnittstelle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMTTP</td>
<td>» 2</td>
<td>real</td>
<td>real</td>
<td>[28]</td>
</tr>
<tr>
<td>ISO, TCP/IP, DNA</td>
<td>» 2</td>
<td>real</td>
<td>real</td>
<td>[60]</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>» 2</td>
<td>real</td>
<td>real</td>
<td>[67]</td>
</tr>
<tr>
<td>DECwindows</td>
<td>» 2</td>
<td>real</td>
<td>real</td>
<td>[144]</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>» 2</td>
<td>real</td>
<td>real</td>
<td>[157]</td>
</tr>
<tr>
<td>Anwender-Schnittstelle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td>» 2</td>
<td>real</td>
<td>real</td>
<td>[8]</td>
</tr>
<tr>
<td>ISO 2a</td>
<td>2</td>
<td>leer</td>
<td>synthetisch</td>
<td>[45]</td>
</tr>
<tr>
<td>FTAM</td>
<td>2</td>
<td>leer</td>
<td>real</td>
<td>[27]</td>
</tr>
<tr>
<td>VMTTP</td>
<td>2</td>
<td>real</td>
<td>real</td>
<td>[29]</td>
</tr>
<tr>
<td>ISO 4</td>
<td>2</td>
<td></td>
<td>synthetisch</td>
<td>[41]</td>
</tr>
<tr>
<td>FTAM, FTP, RPC</td>
<td>2</td>
<td>real</td>
<td>real</td>
<td>[65]</td>
</tr>
<tr>
<td>DECnet</td>
<td>2</td>
<td>leer/real</td>
<td>synthetisch</td>
<td>[69]</td>
</tr>
<tr>
<td>ISO 4</td>
<td>» 2</td>
<td>leer</td>
<td>synthetisch</td>
<td>[70]</td>
</tr>
<tr>
<td>MMS</td>
<td>2</td>
<td>leer</td>
<td>synthetisch</td>
<td>[124]</td>
</tr>
<tr>
<td>ACSE</td>
<td>2</td>
<td>leer</td>
<td>synthetisch</td>
<td>[131]</td>
</tr>
<tr>
<td>CASE, MMFS</td>
<td>2</td>
<td>leer</td>
<td>synthetisch</td>
<td>[146]</td>
</tr>
<tr>
<td>ISO 2b, ISO 4</td>
<td>2</td>
<td>real</td>
<td>synthetisch</td>
<td>[156]</td>
</tr>
<tr>
<td>IP, Mach RPC,</td>
<td>2</td>
<td>leer</td>
<td>synthetisch</td>
<td>[164]</td>
</tr>
<tr>
<td>NFS</td>
<td>2</td>
<td>leer</td>
<td>real</td>
<td>[189]</td>
</tr>
<tr>
<td>ISO 2b, ISO 4</td>
<td>2</td>
<td></td>
<td>synthetisch</td>
<td>[192]</td>
</tr>
<tr>
<td>BDTP</td>
<td>2</td>
<td></td>
<td>real</td>
<td>[16]</td>
</tr>
<tr>
<td>System intern</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP/IP</td>
<td>1</td>
<td>leer</td>
<td>real</td>
<td>[11]</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>1</td>
<td>real</td>
<td>real</td>
<td>[35]</td>
</tr>
<tr>
<td>TCP</td>
<td>1</td>
<td>real</td>
<td>real</td>
<td>[32]</td>
</tr>
<tr>
<td>Raidcomm</td>
<td>1</td>
<td></td>
<td>real</td>
<td>[128]</td>
</tr>
<tr>
<td>SINEC AP</td>
<td>1</td>
<td>leer/real</td>
<td>real</td>
<td>[136]</td>
</tr>
<tr>
<td>PUP, VMTTP</td>
<td>1</td>
<td></td>
<td>real</td>
<td>[145]</td>
</tr>
<tr>
<td>ISO 3</td>
<td>» 2</td>
<td></td>
<td>synthetisch</td>
<td>[42]</td>
</tr>
</tbody>
</table>

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 Programmschnittle 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 ausgearbeiteten Programmteile \( P_1 \)

- 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 übertragenden 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 Paketinhalt nicht akzeptabel ist. Weitere veröffentlichte Meßkonzepte, die Einsicht in systeminterne Vorgänge erlauben, stützten sich auf das Monitorkonzept, wobei verteilte Monitorkomponenten über ein eigenes Monitoretz 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 Rechnersystem, 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, die von der ISO definierte OSI-Leistungsmesskonzept und das im Internet angewandte Leistungsmesskonzept, auf ihre Verwendbarkeit für die hier gestellten Anforderungen untersucht. Darüberhinaus wird das Monitorkonzept² in seinen verschiedenen Varianten den standardisierten Konzepten gegenübergestellt.


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 verteilten 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 Statistik interpreter 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 Zugriffs synchronisation) [81]. Hier wird unter einem Monitor ein Prozeß zur Extrahierung von sich ändernder Information aus einem laufenden System verstanden.

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


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 Maßgrößen 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).“

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


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 Zeittauer (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;

- **Zeittauer** das Zeitintervall zwischen zwei Ereignissen, d.h. ein Abschnitt auf der Zeitachse.


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 Zählerwerts 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


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

Die erste Methode verlangt, daß alle Punkte, die zur Erkennung von Änderungen der Systemakti- vität 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.

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].


- Hardware-Tracing: \(\times 1\)
- Mikrokodemodifikation: \(\times 10\)
- Simulation: \(\times 1000\)
• 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 führt 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 Endereignissen 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 mitberfaßte Zustandsänderungsbedingung erlaubt im nachhinein das Ablaufgeschehen nachzuvollziehen.


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 Endereignis auf das Beginnereignis folgen muß. Jedes zu früh oder verspätet erkannte Endereignis 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.


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.

[Diagramm]

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-Sequencers 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-initiiertes 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 Programmssequenz 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,


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.


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-
<table>
<thead>
<tr>
<th>Ebene</th>
<th>Zeitauflosung</th>
<th>typische Anwendung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mikroprogramm</td>
<td>1 ns</td>
<td>Mikrokode-Optimierung</td>
</tr>
<tr>
<td>Maschinenprogramm</td>
<td>10 ns</td>
<td>Maschinenkode-Optimierung</td>
</tr>
<tr>
<td>Hochsprache</td>
<td>100 ns</td>
<td>Optimierung eines Kodegenerators</td>
</tr>
<tr>
<td>Block</td>
<td>100ms...1ms</td>
<td>Laufzeitmessung</td>
</tr>
<tr>
<td>Programm</td>
<td>1ms...1s</td>
<td>Laufzeitmessung</td>
</tr>
<tr>
<td>Prozeß</td>
<td>1ms...1s</td>
<td>Prozeßverwaltung im Betriebssystem</td>
</tr>
<tr>
<td>System</td>
<td>1ms...1s</td>
<td>interaktive Anwendungen</td>
</tr>
</tbody>
</table>

Tabelle 2: Meßebenen und deren zeitliche Auflösung


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.

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.


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 schrittweise 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 $L_{\text{virt}}$ nur vom Verhältnis der Einschreibfrequenz $F_{\text{Ein}}$ zur Auslesefrequenz $F_{\text{Aus}}$ und der realen Länge des Speichers $L_{\text{real}}$ ab:

\[
L_{\text{virt}} = \left( \frac{F_{\text{Ein}}}{F_{\text{Ein}} - F_{\text{Aus}}} \right) \cdot L_{\text{real}}
\]

(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.


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].


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


![Diagramm von Systemmanagement Applikationsprozeß (AGENT) und Systemmanagement Applikationsprozeß (MANAGER) über CMIP.

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.

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


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


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.


### 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.


![Diagramm](image)

Bild 2.11: Internet Managementmodell


### 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].


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.


Ublicherweise 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 Prozedurauf- und Rücksprungstellen während des Übersetzungsprozesses [181] oder an Verzweigungsstellen eines Programms [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.


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 fest. Die extrahierbare Information und damit das Informationsvolumen wird durch den Meßpunkt festgelegt, aber auch durch den Abgriffsort: Meßfühler, die am Systembus ange schlossen 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 Objektsystems,
- Abspeichern in ein integriertes Speichersystem (gateway)
• Übertragung über das Netz auf ein separates System zur Abspeicherung oder
• Übertragung über separates Monitornetz auf ein separates System zur Abspeicherung.


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 Mesphase, so muß die angesammelte Information an anderer Stelle gesichert werden, bevor der belegte Speicher an das System zurückgegeben werden kann. Ist die Mesphase kürzer als die Lebensdauer des zu messenden Prozesses, so muß ein Zugriff auf dessen Speicherbereich oder eine Transferrmöglichkeit für die Mesdaten möglich sein [204].


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].


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 (1),
- Meßfühler, die an Hardware-Meßpunkten Rückschlüsse auf Objekt-Software erlauben (2) und
- Rückkopplungen (3), 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 einigen 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 Meßführers entsprechen. Differenziertere Meßfühler lassen eine Datenreduktion durch Maskierung einzelner Meßpunkte zu: entweder wird die Meßpunktserkennung 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.


2.3.2 Software-Monitor


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 anstoßt, 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 Befehlschnellspeichern (engl. Command Caches), in Betriebssystemen mit Kachelspeicher kann dies Einfluß auf die aktuell im Speicher befindlichen Seiten und somit Seitenwechsel zur Folge haben.


<table>
<thead>
<tr>
<th>Art</th>
<th>Laufzeit (µs)</th>
<th>=STORE-Befehle</th>
</tr>
</thead>
<tbody>
<tr>
<td>inaktiver Sensor</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>ausgefilterter Sensor</td>
<td>90..165</td>
<td>45..80</td>
</tr>
<tr>
<td>aktiver Sensor</td>
<td>600..1400</td>
<td>85..200</td>
</tr>
</tbody>
</table>

Tabelle 3: Laufzeiten bei Monitorprozeduren nach [190]

In Tabelle 3 sind typische Laufzeitwerte einer Mikroprogramm-kodierten Monitorprozedur aufgeführt, Versionen mit Prozeduraufrißen 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 größere 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 Abcsätzung an, wieviele Intervalle mindestens gemessen werden müssen, um signifikante Nachkomma-stellen (bei vorgegebener Uhrenauflösung) angeben zu können (vergl. Tabelle 4).

<table>
<thead>
<tr>
<th>zu messende Intervaldauer</th>
<th>2 sign. Nachkomma-stellen</th>
<th>3 sign. Nachkomma-stellen</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 µs</td>
<td>800.000</td>
<td>80.000.000</td>
</tr>
<tr>
<td>100 µs</td>
<td>80.000</td>
<td>8.000.000</td>
</tr>
<tr>
<td>1 ms</td>
<td>7.000</td>
<td>700.000</td>
</tr>
<tr>
<td>10 ms</td>
<td>400</td>
<td>40.00</td>
</tr>
</tbody>
</table>

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],
- Objekttprogramm-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] müßte zur Vermeidung von Überschreibungen der Puffer periodisch ausgelesen werden, weshalb die Messung für die Dauer des Datentransfers unterbrochen wird.

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

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).

![Diagramm des Hybrid-Monitors](image)

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 (1) einem externen Hardware-Monitor zur Übernahme (2) 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 (3) 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-Meißfühlern besteht. Tabelle 6 gibt einen Überblick über erzielte Werte.

<table>
<thead>
<tr>
<th>HW-Meißfühler</th>
<th>Monitorregister</th>
<th>Dauer für Registerzugriff</th>
<th>zeitliche Auflösung</th>
<th>Literatur</th>
</tr>
</thead>
<tbody>
<tr>
<td>nein</td>
<td>8 bit parallel, E/A-Schnittstelle</td>
<td>8 μs</td>
<td></td>
<td>[42]</td>
</tr>
<tr>
<td>nein</td>
<td>2’16 bit parallel, Speicherzelle</td>
<td>4 μs</td>
<td>1...100 μs</td>
<td>[68]</td>
</tr>
<tr>
<td>ja</td>
<td>8 bit parallel, E/A-Schnittstelle</td>
<td>5 μs</td>
<td>100 ns</td>
<td>[82]</td>
</tr>
<tr>
<td>nein</td>
<td>16 bit parallel, Speicherzelle</td>
<td>3...8 μs</td>
<td>1 μs</td>
<td>[142]</td>
</tr>
<tr>
<td>ja</td>
<td>32 bit, Speicherzelle</td>
<td>1 μs</td>
<td>1 μs</td>
<td>[183]</td>
</tr>
</tbody>
</table>

Tabelle 6: Charakteristische Größen einiger Hybrid-Monitore

Bei all diesen Hybrid-Monitoren wurde das Speicherproblem in ähnlicher Weise gelöst: die von den Meißfühlern 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


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 T2 im Vergleich zu T1 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.
Können die Meßpunkte in verteiltten Objektsystemen sowie deren Auftrittszeitpunkte erkannt werden, so besteht die Möglichkeit, die Einwegverzögerung direkt zu messen, entsprechend Bild 3.2b).

Die fehlerrächtige Halbierung der Auftragsbearbeitungszeit $T_A$ aufgrund der nicht einzel zu messbaren Hinaufzeit $T_H$, der geschätzten Bearbeitungszeit $T_B$ und der Rücklaufzeit $T_R$ entfällt. Darüber hinaus 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


![Diagramm der Synchronisationsmöglichkeiten](image)

**Bild 3.3:** Klassifizierung der Synchronisationsmöglichkeiten


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

<table>
<thead>
<tr>
<th>T [in Sekunden]</th>
<th>P(Fehler ≥ T) ohne Filterung</th>
<th>P(Fehler ≥ T) mit Filterung</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>0.98</td>
<td>0.93</td>
</tr>
<tr>
<td>0.01</td>
<td>0.53</td>
<td>0.079</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>= 0</td>
</tr>
<tr>
<td>1</td>
<td>0.002</td>
<td>= 0</td>
</tr>
<tr>
<td>10</td>
<td>0.0007</td>
<td>= 0</td>
</tr>
</tbody>
</table>

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.


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)),
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.


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.

<table>
<thead>
<tr>
<th>Verfahren</th>
<th>Zeitverlauf</th>
<th>Zeitzuordnungsvariante nach Bild 3.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>meldungsbasiert</td>
<td>Zeitsprünge (vor- und rückwärts)</td>
<td>b) und d)</td>
</tr>
<tr>
<td>Takt-Leitung</td>
<td>stetig</td>
<td>c)</td>
</tr>
<tr>
<td>Software-Protokolle</td>
<td>stetig Zeitsprünge (vorwärts)</td>
<td>d)</td>
</tr>
<tr>
<td></td>
<td>Zeitsprünge (vor- und rückwärts)</td>
<td></td>
</tr>
<tr>
<td>logische Ordnung</td>
<td>Zeitsprünge (vorwärts)</td>
<td>e)</td>
</tr>
<tr>
<td>Resynchronisationspunkte</td>
<td>stetig</td>
<td>e)</td>
</tr>
</tbody>
</table>

Tabelle 8: Zeitverläufe bei unterschiedlichen Synchronisationsverfahren

Problematisch bei den Verfahren, die die Zeitinformation verteilten, 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 schrittweise 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 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 Monitoretz 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 netzweise 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.

<table>
<thead>
<tr>
<th>Verbindungsschema</th>
<th>Vorteile</th>
<th>Nachteile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor integriert in Objektsystem</td>
<td>nur 1 Netz notwendig, einfach</td>
<td>Einfluß auf Objektsystem, Einfluß auf Netz</td>
</tr>
<tr>
<td>Monitor verwendet vorhandenes Netz</td>
<td>kein Einfluß auf Objektsystem, nur 1 Netz notwendig</td>
<td>aufwendig, Einfluß auf Netz</td>
</tr>
<tr>
<td>spezielles Monitornetz</td>
<td>kein Einfluß auf Objektsystem, kein Einfluß auf Netz</td>
<td>aufwendig, 2 Netze notwendig</td>
</tr>
</tbody>
</table>

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.


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

Das Objektverhalten 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.


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.

Bezuglich 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.
3.6.1 Meßvorbereitung

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

- Verfügbarkeit der verteilten Meßkomponenten überprüfen,
- verteilte Meßkomponenten initieren (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 implementierungssabhängig 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.

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


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-Philosophie alle Ebenen der Fertigung umfaßt. Daher scheint die zusätzliche Verlegung eines Monitornetzes, aber auch eines separaten Synchronisationsnetzes am Platz. 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 scheint aus, da Messungen bis in die Anwendungsebene durchzuführen sind.

4.1 Hybridmonitor


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 Forde rung nach einer Minimallösung widerspricht. Auch das Verfahren nach Bild 3.4d belastet durch Software-Synchronisationsprotokolle das vorhandene Netz und scheint 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 zwi schengespeichert 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 Auswerteestation übertragen. Andererseits können durch sofortiges Auslesen der dezentralen Speicher die Meßdaten für eine laufende Auswertung an die Auswerteestation transferiert werden, wenn die daraus resultierende Netzbelastung akzeptiert wird.


![Diagram](image)

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

Bei erhöhtem Aufwand sind neben der im Bild 4.1 gezeigten Verbindungsstruktur aller Monitorkomponenten über das vorhandene Netz die in Bild 3.5 gezeigte Verbindung über ein zusätzliches separates Monitoretz 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 Monitoretz direkt angeschlossen sind.
Bild 4.2: Hybrid-Monitor mit unterstützendem Monitornetz (SUT Objektsystem, M Monitorkomponente, AS zentrale Auswertestation)

Als Minimalkonfiguration werden die Objektsystem-abhängigen Messfühler sowie die externen Monitorkomponenten genutzt, 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 Diskettspeicher.

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.


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 Abgriffsstellen 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.


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-Meißfühler 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-Meißfühler

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


4.1.1.2 LAN-Koprozessor

Alle erhaltenen 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-Koprozes- sor eine Kopie seiner internen Register im gemeinsamen Speicher ab (Bild 4.4b) [92],
- Zweiter-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.


- 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 Rah- mens empfangen wird.

Zu diesen Zeitpunkten kann aber noch nicht der korrekte Empfang oder Versand eines Rahmens festgestellt werden, weshalb für die hier gewählte Meßumgebung eines Lokalen Netzes als Meß- punkt 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 Adresse- und 16bit oder 32bit Datenpfad, über den alle Baugruppen angeschlossen sind. Dabei gibt es unterschiedliche Zugriffsmodi:

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


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 Speicherlöchern, für die eine Adreßdekodierung vorhanden ist oder
- Einbau eines spezifischen Monitoranschlusses.


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 nirgendwo 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.

![Diagramm Informationsfluß im Bussystem-Meßfühler](image)

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 Adresse 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


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.


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.


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 Meßfühlern,
- 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-Speichern 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-
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


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


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).
Neben der Überwachung der einzelnen Funktionseinheiten ist die Bereitstellung von Kommunikationsdiensten die Hauptaufgabe der Steuerung. Alternativ zu einer lokal angeschlossenen Bedienerkonsole kann von der Auswerteinheit 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 Auswerteinheit 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 fehler tolerantes 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).

<table>
<thead>
<tr>
<th>Type</th>
<th>Class I</th>
<th>Class II</th>
<th>Class III</th>
<th>Class IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

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

![Diagramm](image)

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).


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 Taktpériode 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 Umschaltszeit beinhaltet und die Anzahl abgefragter Meßfühler je Messung konstant ist, so berechnet sich die Erfassungszeit \( T_E \) zu:

\[
T_E = \left[ T_{Eu}, T_{Eo} \right] \\
T_{Eu} = (ZS \cdot n + ID) \cdot T_M \\
T_{Eo} = ((ZS + 1) \cdot n + ID) \cdot T_M
\]

(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 Transferrmeldung 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 \textit{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 Bouter, 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.


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

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 Ereignissspuren zu der globalen Ereignisspur berücksichtigt.


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 scheitert 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:

\[
F^i = (T^i_1, T^i_2, T^i_3 \ldots )
\]  

(GL.3)
Bild 4.10: Zuordnung der SYNC-Ereignisse über Zeitabstände

Da aufgrund des Abfragebetriebes 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) 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:

$$c(l) = \frac{\sum_{k} (T^i_k - T^i_l)(T^j_k + 1 - T^j_l)}{\sqrt{\sum_{k} (T^i_k - T^i_l)^2 \sum_{k} (T^j_k + 1 - T^j_l)^2}}$$ (Gl.4)

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

Dort, wo der Korrelationskoeffizient maximal ist, sind die Folgen am ähnlichsten und bei $c=l$ 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 Periode 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 derSYNC-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 beschriebenen Verfahren bearbeitet werden können.
Zusätzlich lassen sich zwei Ansätze (SYNC-Abstände und -Inhalte) kombinieren, 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.


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 1 Zuordnung der SYNC-Rahmen,
Schritt 2 Bildung der Referenzspur aus den lokalen Spuren und
Schritt 3 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 Zeitleiste 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 Ereignisfeldern entfernt und diese zeitlich in die globale Ereignisspur eingesortiert. Gleichzeitig findet eine Kennzeichnung der einzelnen Ereignisse statt, so daß bei der späteren Auswertung der Herkunftsdatum erkennbar bleibt. Darüberhinaus kann eine Anpassung einzelner Ereignisfelder erfolgen, damit die Auswertungsprogramme mit maschinengeeigneten Datenformaten arbeiten können.

4.4 Meßdatenauswertung


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 vorgebar 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.


```
<table>
<thead>
<tr>
<th>ID</th>
<th>Info_1</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Prozess</th>
<th>Name</th>
<th>Priorität</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Bild 4.13: Grobmustern 1 und 2 mit Teilfeldern


Das Feinmuster ProzedurEnde basiert auf Grob muster 1 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

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


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 Grobmustern 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.

```
1 { ID[2], var; Info_1[6], var; Parameter[8], var; }
2 { Prozess[4], fix=6; Name[8], var; Priorität[4], var; }

Teilfeld (variabel)
Teilfeld (konstant)
Lfd. Nummer für Grob muster
```

Die erste Zeile beschreibt Grob muster 1, die zweite Zeile das Grob muster 2 aus Bild 4.13. Der Verfeinerungsschritt entsprechend Bild 4.14 folgt mit:

```
ProzedurStart 1 { ID=0; Info_1=0x30; }
ProzedurMitte 1 { ID=1; }
ProzedurEnde 1 { ID=0; Info_1=0x31; Parameter=0.12; }
ProzedurEndeOK 1 { ID=0; Info_1=0x31; Parameter=0; }

Verfeinerung des Grobmusters
Name des Feinmusters
```

und für das verkettete Grob muster nach Bild 4.15:

```
Prozedur 1 + 2 { ID=0; Info_1=0x30; Priorität=2; }
```

Mit diesen Eingabedefinitionen kann eine Erkennung der zugehörigen Ereignisse in der Ereignis spur 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 Datenbeschreibungs sprache 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 $\delta$ 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: $A(k) = P\{A \leq k\} = \sum_{i=0}^{k} a_i$

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

Die aus einer diskreten Verteilung berechenbaren statistischen Kenngrößen sind:
Erwartungswert: \[ E[h(k)] = \sum_{k=0}^{\infty} h(k)a_k \]

1. tes gewöhnliches Moment: \[ m_i = E[A^i] = \sum_{k=0}^{\infty} k^i a_k \]

1. tes zentrales Moment: \[ \mu_i = E[(A - m_1)^i] = \sum_{k=0}^{\infty} (k-m_1)^i a_k \]

Mittelwert: \[ E[A] = m_1 \]

Varianz: \[ \text{VAR}[A] = \mu_2 \]

Standardabweichung: \[ \sigma_s = \sqrt{\text{VAR}[A]} \]

Variationskoeffizient: \[ c_s = \frac{\sigma_s}{m_1} \]

4.4.2.2 Meßauswertung

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


- Bei der Typstatistik werden alle Leistungsgrößen nur für ein Feinmuster berechnet. Beginn und Ende eines Intervalls werden durch gleiche Feinmuster definiert.
- 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 einzuhaltenenden 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-Medienzugriffssverfahren, ausgewählt. Der auf diesem Netz messbare Verkehr entspricht den in [61] gemessenen Verkehren einer Entwicklungsumgebung.


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.

<table>
<thead>
<tr>
<th></th>
<th>WS20</th>
<th>Intel310</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT/RTX86</td>
<td>IHI</td>
<td></td>
</tr>
<tr>
<td>RTX286</td>
<td>IHI</td>
<td></td>
</tr>
<tr>
<td>iRMX86</td>
<td></td>
<td>MIP</td>
</tr>
<tr>
<td>iRMX II</td>
<td></td>
<td>MIP/IHI</td>
</tr>
<tr>
<td>XENIX V</td>
<td>IHI</td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 11: Kommunikationsschnittstellen bei verschiedenen Betriebssystemen

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.


Ein direkter Zugriff 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.

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


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.


Für den Meßfühler bedeutet dies, daß mit den Signalen CA und INT wichtige Zustände erkannt werden können. Ob der LAN-Koprozessor Netzdten empfangen hat, läßt sich aus den Steuerleitungen zum Netzan schlüßkabel 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 miteinflugt wird. Damit sind als Beobachtungsstellen dieser Netzzugangsmesßfühler folgende Signalleitungen des LAN-Koprozessors wichtig:

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

Die Netzan schlüßleitung CRS (Carrier Sense) signalisiert den Empfang von Netzdten. 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


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.

Bild 5.5: Zeitverhalten am Meßfühler

Im Falle eines Sendeversuchs (1 bis 5) nach Bild 5.5 durchläuft der Automat seine Zustände, bis er zum Zeitpunkt 6 erkennt, ob der Transfer erfolgreich war oder nicht. Einen Rahmenempfang (6 bis 9) erkennt der Automat durch aktives CRS ohne vorangehendes CA. Beim Zugriff auf den SCB (9) 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.


<table>
<thead>
<tr>
<th>Signal</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>RxD</td>
<td>serielle Empfangsdaten (engl. Receive Data)</td>
</tr>
<tr>
<td>RxC</td>
<td>Empfangstakt (engl. Receive Clock)</td>
</tr>
<tr>
<td>CRS</td>
<td>Empfangsanzeige (engl. Carrier Sense)</td>
</tr>
<tr>
<td>COL</td>
<td>Kollisionsanzeige (engl. Collision)</td>
</tr>
</tbody>
</table>

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

Die Messmöglichkeiten, die mit unterschiedlichen Messfühlern an diesem Meßpunkt realisiert sind, zeigt Bild 5.6.

![Diagramm](image)

**Bild 5.6:** Meßmöglichkeiten mit rückgekoppeltem Schieberegister


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 Octets (Rahmen inklusive Prüfambel) 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.
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:

\[ n = (16 \text{ bit, 32 bit, 48 bit, 64 bit}). \]

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,


![Diagramm zur VLSI-Baustein 48 bit Mustererkennung](image)

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.


![Format eines Ereigniseintrags des Meßfühlers am Netzzugang](image)

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 Speicherzügen 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


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:

1. Version ohne Adreßfilter und nur ein FIFO (16bit breit) [62],


<table>
<thead>
<tr>
<th>Busanschluß</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ladbare Adreßreferenz</td>
<td>WS20</td>
<td>WS20 / IBM-AT</td>
<td>WS20 / IBM-AT</td>
</tr>
<tr>
<td>Adreßfilterung</td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>ladbare Masken</td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>integrierte Meßfühler</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Breite des FIFO-Speichers</td>
<td>16 bit</td>
<td>16 bit</td>
<td>18 bit</td>
</tr>
<tr>
<td>Anschluß für Meßfühler LAN-Koprozessor</td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Infobreite für Meßfühler LAN-Koprozessor</td>
<td>4 bit</td>
<td>18 bit</td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 13: Unterschiedliche Systembus-Meßfühler

5.1.3 Integration aller Meßfühler


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 passender 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.

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


Neben Anwendungsprogrammen kennt QOS Treiberprogramme, die die Verwaltung der Schnittstellen (Konsolenanschluß über V.24, Locales 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:

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.

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 ($DO..15$) 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.

![Diagramm der Initialisierungsphase](attachment:diagram.png)

**Bild 5.13: Initialisierungsphase (Zyklusgröße 8)**


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:

\[ T^n_f = \frac{2^{32} \cdot 4 \cdot n \cdot f}{F} \]  

(Gl. 5)

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


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

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 Transportstanz 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 fehltolerantes, auf LLC Class I basierendes verbindungsähnliches Protokoll mit Quittierungsmechanismus auch bei Gruppenadressierung [153].


```
1 oct. 1 oct. 1 oct. 1 oct. 0...512 oct.
DA_AP  SA_AP  SEQ  MSGTYP  DATA

1 oct. 1 oct. 1 oct.
DSAP  SSAP  CONTROL  DATA

7 oct. 1 oct. 6 oct. 6 oct. 2 oct. 46...1500 oct.
Preambel SFD DA SA LENGTH DATA

4 oct. FCS
MAC (CSMA/CD)
```

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


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, wie viele Quittungen er erwartet. Werden alle Quittungen innerhalb eines vorgebaren Zeitraumes empfangen, so wird dieser Transfer erfolgreich beendet. Fehlen einzelne Quittungen, so bricht der Versuch wieder nach einer vorgebaren 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.


### 5.4.3 Befehlsatz

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).

<table>
<thead>
<tr>
<th>Phase</th>
<th>Befehl</th>
<th>Kodierung (Befehl, Meldung)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialisierung</td>
<td>ABORT, INIT</td>
<td>a, A</td>
</tr>
<tr>
<td>Messung</td>
<td>PROGFREQ, MEASURE, STATUS</td>
<td>p, P, m, M, q, Q</td>
</tr>
<tr>
<td>Transfer</td>
<td>TRANSMIT, RETRANSMIT</td>
<td>t, T, r, R</td>
</tr>
</tbody>
</table>

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 ;

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


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. Ereignis-
nspuren) 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 In-
tervalls 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 dynamischen Intervalls innerhalb eines statischen Intervalls und Wiederhol-
ung des gesamten Vorgangs bei verschobenem dynamischen Intervall (Bild 5.16) bedeutet, daß
viele Korrelationskoefﬁzienten berechnet werden. Der Benutzer kann über die Steuerdatei alle Pa-
rameter zur Korrelationsberechnung vorgeben. Dies verlangt vom Benutzer oft zu viel Detail-
wissen, 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.

<table>
<thead>
<tr>
<th>korreliere SPUR.001 mit SPUR.003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gesamt-Ergebnis:</td>
</tr>
<tr>
<td>-2  6  50.0%</td>
</tr>
<tr>
<td>-1  3  25.0%</td>
</tr>
<tr>
<td>1  1  8.3%</td>
</tr>
<tr>
<td>3  2  16.7%</td>
</tr>
<tr>
<td>---&gt;Wahl: -2</td>
</tr>
</tbody>
</table>

Bild 5.17: Beispielergebnis der Korrelation zweier Ereignisspuren
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 $[T_n-d, T_n+d]$ beschreibt den Zeitbereich, innerhalb dessen ein Zeitstempelwert gleichverteilt liegen könnte. Einfluß auf die Zeitabweichung $d$ haben die freilaufenden Uhren ($\Delta z$), die Verzögerungen in den Meßfühlern und die 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.

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:

\[
\text{INTERPOLATION [linear | akima | spline ]}
\]
\[
\text{Art der Zeitinterpolation zwischen zwei SYNCS}
\]

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:

\[
P_i(x_i) = A_i + B_i(x-x_i) + C_i(x-x_i)^2 + D_i(x-x_i)^3
\]

mit den Koeffizienten:

\[
A_i = y_i
\]
\[
B_i = t_i
\]
\[
C_i = \frac{3m_i - 2t_i - t_{i+1}}{x_{i+1} - x_i}
\]
\[
D_i = \frac{t_i + t_{i+1} - 2m_i}{(x_{i+1} - x_i)^2}
\]
\[
m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}
\]
\[
t_i = \frac{|m_{i+1} - m_i|^3 m_{i-1} + |m_{i-2} - m_{i-1}|^3 m_i}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|}
\]

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.

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: 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 Ausgangsspurr 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: \[ g = (s \times \text{MAX}_{\text{MF}}) + 1 \quad \text{mit} \quad g \in (0...2^{8}-1) \quad (\text{Gl.8}) \]

\text{MAX}_{\text{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 (\text{MAX}_{\text{MF}}=4) bzw. 32 (\text{MAX}_{\text{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 Ableitungsmethoden (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:

\[
( \text{Informationsfeld UND FeldMaske} ) \text{ EXOR WertMaske} = 0 
\]  
(Gl.9)

Die Masken ergeben sich aus der Feinmusterdefinition von \textit{ProzedurStart} nach Bild 4.30:

\[
\begin{align*}
\text{ProzedurStart:} & \quad \text{FeldMaske} = 11\quad 11111\quad 00000000b = 0xff00 \\
& \quad \text{WertMaske} = 00\quad 110000\quad 00000000b = 0x2000 \\
\end{align*}
\]

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.


Eine weitere Möglichkeit zur Histogrammbildung mit quasi-logarithmischer Skala besteht, wenn man das Intervall zwischen Minimal- und Maximalwert in 2¹⁶ Abschnitte aufteilt. Als obere Klassenspanne \( K_i \) dient folgender Wert:

\[
K_i = \text{Min} + K_{\text{Schrit}} \times 2^i \quad (0 \leq i \leq 16) \quad \text{(Gl.10)}
\]

und

\[
K_{\text{Schrit}} = \frac{\text{Max} - \text{Min}}{2^{16}} \quad \text{(Gl.11)}
\]

Gegenüber der ersten Möglichkeit liefert diese Berechnung immer 17 Klassen (Min sowie \( K_i \) als oberen Klassenschritt), verlangt aber zwei Durchläufe, da Minimal- und Maximalwert zur Berechnung der Schrittweite \( K_{\text{Schrit}} \) bekannt sein müssen. Vorteilig an der logarithmischen Skaleneinteilung ist, daß die Berechnung im Programm mit einfachen Schiebebefehlten (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 Aufzeitunterschiede mehrere Größenordnungen betragen (PStop→PStart).

5.6.2.2 Statistische Auswertung bezüglich Ereignispaaren

Zur Abstands messung 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:
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 Abhängigkeitssätze denkbar:

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

Bild 5.22: Logarithmische gegenüber linearer Klasseneinteilung
(links: PStart→PStop und rechts: PStop→PStart, nach Kap. 7.1.2)
Bild 5.23: Paarbildung Muster1→Muster2 bei Mehrdeutigkeit der Musterdefinitio-
nen (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

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


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

![Bild 5.25: Zuordnung bei Musterserien](image)

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


$$T_{K7} = 7 \times t_{Cl}$$  \hspace{1cm} \text{(Gl.12)}

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


$$T_{Rahmen} = t_{Cl} \times 72 \times 8$$  \hspace{1cm} \text{(Gl.13)}

$$T_{K6} = 6 \times t_{Cl}$$  \hspace{1cm} \text{(Gl.14)}

$$T_{Abstand} = 96 \times t_{Cl}$$  \hspace{1cm} \text{(Gl.15)}

Aus (Gl.13-15) summiert sich die Zeit bis zum Zugriff auf den SCB bei minimal langen Rahmen:
\[ T_{\text{SCB}} = T_{\text{Abstand}} + T_{\text{Rahmen}} + T_{k6} \]  

(Gl.16)


![Diagram of bus position and frame synchronization]

\textbf{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.

\textbf{Annahme:}

\textbf{Empfangen:} Interne FIFO-Puffer bei Rahmenende belegt mit Daten (schlechterer Fall)

  - 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.

\[
Zeit = \text{Buszuteilung} + 8 \text{ Schreibzyklen (evtl. mehrere DMA-Bursts)}
+ \text{Buszuteilung} + 3 \text{ Schreibzyklen}
+ \text{Buszuteilung} + 1 \text{ Lesezyklus}
+ \text{Buszuteilung} + 2 \text{ Schreibzyklen}
\]

- 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.

\[
Zeit = \text{Buszuteilung} + 1 \text{ Schreibzyklus}
+ \text{Buszuteilung} + 1 \text{ Schreibzyklus}
\]

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


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

Für den Einschreibeveorgang wird für die Dauer $T_{\text{Fifo}} = 200 \text{ ns}$ 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 $T_D$:

\[
T_D = t_{\text{WLZ}} + t_{\text{WE}} + t_A \quad \text{(Gl.17)}
\]

mit:
- $t_{\text{WLZ}}$: Aktivierung der Ausgänge (min. 20 ns, max. 40 ns)
- $t_{\text{WE}}$: 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 Ausleesezeitpunkt:

\[
T_{\text{RE}} = T_{\text{K7}} + T_{\text{Fifo}} + T_D \quad \text{(Gl.18)}
\]
\[
T_{\text{SR}} = T_{\text{SCB}} + T_{\text{Fifo}} + T_D \quad \text{(Gl.19)}
\]

Alle einzelnen Verzögerungswerte sind in Tabelle 16 zusammengefaßt.
<table>
<thead>
<tr>
<th>Name</th>
<th>Zeit</th>
</tr>
</thead>
<tbody>
<tr>
<td>TK6</td>
<td>600 ns typ.</td>
</tr>
<tr>
<td>TK7</td>
<td>700 ns typ.</td>
</tr>
<tr>
<td>TFifo</td>
<td>200 ns typ.</td>
</tr>
<tr>
<td>TD</td>
<td>220 ns max.</td>
</tr>
<tr>
<td>TRahmen</td>
<td>≥ 57600 ns min.</td>
</tr>
<tr>
<td>TAbstand</td>
<td>≥ 9600 ns min.</td>
</tr>
<tr>
<td>TSCB</td>
<td>≤ 1230400 ns max.</td>
</tr>
<tr>
<td>TRE</td>
<td>≤ 1120 ns</td>
</tr>
<tr>
<td>TSR</td>
<td>≥ 67200 ns min.</td>
</tr>
</tbody>
</table>

Tabelle 16: Verzögerungen im Hardware-Meißfühler

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 TK6 nach Rahmenende der erfolgreiche Transfer des vorherigen Rahmens gemeldet, bevor im nächsten Takt (TK7) das SYNC-Ereignis des aktuellen Rahmens signalisiert wird.


### 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 frühsten möglichen Abfrage aus dem FIFO-Baustein ergibt sich unter Verwendung von (Gl.17) zu:

\[ T_{\text{Bus}} = T_D \]

(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) ge- strichen werden darf (entspricht einer Verschiebung der Zeichse 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 schnellsstmögliche 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 Kodebeispeilen aus. Dazu sind folgende einfache Sequenzen zur Wertbildung samt dem folgenden Schreibzugriﬀ gemessen worden:

```c
*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.

<table>
<thead>
<tr>
<th>Operand</th>
<th>Assemblerkode</th>
<th>Takte nach [91]</th>
</tr>
</thead>
<tbody>
<tr>
<td>fix</td>
<td>mov es,$T2000, les bx,DWORD PTR es::_ZeigerAufMonitorRegister, mov bx, [bx]+205</td>
<td>19</td>
</tr>
<tr>
<td>variabel</td>
<td>mov es,$T2000, les bx,DWORD PTR es::_ZeigerAufMonitorRegister, mov ax, [bp-4], mov es::[bx], ax</td>
<td>19</td>
</tr>
<tr>
<td>ODER-verknüpft</td>
<td>mov es,$T2000, les bx,DWORD PTR es::_ZeigerAufMonitorRegister, mov ax, [bp-6], or ax, [bp-8], mov es::[bx], ax</td>
<td>19</td>
</tr>
</tbody>
</table>

Tabelle 17: Maximale Taktpetioden 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-Speichermode

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

Wenn eine Meßzeitabweichung von $\Delta Z_i$ 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 $\Delta Z_i$ aufsummiert hat.

$$m \leq \frac{\Delta Z_i}{t \cdot \left( \frac{\Delta F}{F} \right)}$$  \hspace{1cm} (Gl.21a)

Daraus ergibt sich dann die minimale Resynchronisationsrate $R_z$:

$$R_z = \frac{F}{m} \geq \frac{\left( \frac{\Delta F}{F} \right)}{\Delta Z_i}$$  \hspace{1cm} (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 erzielbar.

<table>
<thead>
<tr>
<th>Typ</th>
<th>$\pm 5$ ppm</th>
<th>$\pm 2$ ppm</th>
<th>$\pm 1$ ppm</th>
</tr>
</thead>
<tbody>
<tr>
<td>unkompensiert (PXO)</td>
<td>Temperatur</td>
<td>Spannung</td>
<td>Alterung</td>
</tr>
<tr>
<td>temperaturkompensiert (TCXO)</td>
<td>$\pm 2$ ppm Temperatur</td>
<td>$\pm 0.1$ ppm Spannung</td>
<td>$\pm 1$ ppm/Jahr Alterung</td>
</tr>
<tr>
<td>temperaturstabilisiert (OCXO)</td>
<td>$\pm 0.1$ ppm Temperatur</td>
<td>$\pm 10^{-8}$ Spannung</td>
<td>$\pm 10^{-8}$ /Jahr Alterung</td>
</tr>
</tbody>
</table>

Tabelle 20: Notwendige Resynchronisationsrate $R_z$ in Abhängigkeit vom Oszillatorotyp und der erlaubten Abweichung $\Delta Z_i$
6.2.2 Synchronisation mittels Netzverkehr

Werden nur solche Rahmen zur Resynchronisation herangezogen, die durch die vorhandene Nutz-kommunikation 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 Fileserven oder Printserven 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 virtualen 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].

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

6.2.3 Synchronisation mittels synthetischem Verkehr


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:

$$T_V^n = \text{random}(0, \min(2^n, 2^{10})) \cdot T_{\text{slot}}$$  \hspace{1cm} (Gl.22)

wobei:  
- $n$: Anzahl bereits erlittener Kollisionen
- $T_{\text{slot}}$: Zeit für doppelte Kanallaufzeit (51,2 µs)
Zu dieser rahmenspezifischen Verzögerungszeit addiert sich die bereits verbrauchte Übertragungszeit $T_H$ sowie die Kollisionserkennungszeit ($T_{JAM}=3.2 \mu s$) und die minimale Wartezeit zwischen aufeinanderfolgenden Rahmen ($T_{GAP}=9.6 \mu s$). Die gesamte Verzögerungszeit summiert sich zu:

$$T_V(k) = T_V^0 + \sum_{n=1}^{k} T_V^{n-1} + T_{JAM} + T_{GAP} + T_V^n$$

(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.


### 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


#### 6.3.2 Berechnungsgenauigkeit im Statistikinterpreter

Das zur Quantilwerteberechnung verwendete heuristische Verfahren von [108] liefert dann sehr genaue Schätzwerte (mittleres Fehlerquadrat $\leq 0.02$), wenn die Stichprobenanzahl mehr als einhundert Ereigniswerte umfaßt und die Klassenverteilung annähernd parabolischen Verlauf annimmt.

<table>
<thead>
<tr>
<th>Klassenverteilung (log2)</th>
<th>Klassenverteilung (exakt log2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>obere Grenze</td>
<td>Anzahl</td>
</tr>
<tr>
<td>7.6928E-04</td>
<td>3</td>
</tr>
<tr>
<td>7.9744E-04</td>
<td>142</td>
</tr>
<tr>
<td>8.2560E-04</td>
<td>0</td>
</tr>
<tr>
<td>8.8192E-04</td>
<td>0</td>
</tr>
<tr>
<td>9.9456E-04</td>
<td>33</td>
</tr>
<tr>
<td>1.2198E-03</td>
<td>0</td>
</tr>
<tr>
<td>1.6704E-03</td>
<td>0</td>
</tr>
<tr>
<td>2.5715E-03</td>
<td>0</td>
</tr>
<tr>
<td>4.3738E-03</td>
<td>0</td>
</tr>
<tr>
<td>7.9782E-03</td>
<td>0</td>
</tr>
<tr>
<td>1.5187E-02</td>
<td>0</td>
</tr>
<tr>
<td>2.9605E-02</td>
<td>0</td>
</tr>
<tr>
<td>5.8441E-02</td>
<td>0</td>
</tr>
<tr>
<td>1.1611E-01</td>
<td>0</td>
</tr>
<tr>
<td>2.3146E-01</td>
<td>0</td>
</tr>
<tr>
<td>4.6214E-01</td>
<td>0</td>
</tr>
<tr>
<td>9.2352E-01</td>
<td>0</td>
</tr>
<tr>
<td>1.8463E+00</td>
<td>339</td>
</tr>
</tbody>
</table>

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 Ereignis spur.

6.4.2 Datenvolumen für Statistikauswertung

Für die Auswertung ist infolge der Trennung von Meßdaten 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:

- Datenbeschreibungdatei
- Referenzdatenbank  
  Referenzierteil je Feinmuster der Datenbeschreibung mit Positionsanzeigen  
  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 Ereignis spur von 0,25 MBytes dazugerechnet werden muß.


6.5 Rechenzeiten

6.5.1 Meßdauer

Die Dauer der Messung hängt von der Erzeugungsraten 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.

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

Dabei gilt:

$f$ Frequenzteiler mit $f \in \{2, 8, 32, 64\}$

$Z$ Zykluslänge mit $Z \in \{4, 8\}$
Einträge im gesamten Meßspeicher mit $L \in (32K, 96K, 160K, 224K)$

a) Anzahl aktiver Meßfühler je Zyklus ($0 < a \leq Z$)

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

$$R_1 = \frac{1}{160 \text{ns} \times f \times Z}$$

(Gl. 25)

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

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

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 enthält die eigene Prüfsummenkontrolle, es wird aber trotz schnellerem Bittakt nur ein Durchsatz von wenigen KBytes/s erreicht.

6.5.2 Korrelationsberechnungen

Zur Spurenuordnung werden die Korrelationsberechnungen in parametergesteuerten Schleifen durchgeführt. Als Vorgabe für die Parametertriplen dienen die Größen $S$ (statisches Intervall) und $D$ (dynamisches Intervall) aus Tabelle 23.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Startwert</th>
<th>Endwert</th>
<th>Inkrement</th>
</tr>
</thead>
<tbody>
<tr>
<td>S (Start)</td>
<td>$S_\text{S}$</td>
<td>$S_\theta$</td>
<td>$S_\text{i}$</td>
</tr>
<tr>
<td>S (Länge)</td>
<td>$LS_\text{S}$</td>
<td>$LS_\theta$</td>
<td>$LS_\text{i}$</td>
</tr>
<tr>
<td>D (Start)</td>
<td>$D_\text{S}$</td>
<td>$D_\theta$</td>
<td>$D_\text{i}$</td>
</tr>
<tr>
<td>D (Länge)</td>
<td>$LD_\text{S}$</td>
<td>$LD_\theta$</td>
<td>$LD_\text{i}$</td>
</tr>
</tbody>
</table>

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

Die Anzahl durchgeführter Korrelationsberechnungen $K$ berechnet sich dann zu:

$$K = \left\lfloor \frac{(S_\text{S} - S_\theta + 1)}{S_\text{i}} \right\rfloor \times \left\lfloor \frac{(LS_\text{S} - LS_\theta + 1)}{LS_\text{i}} \right\rfloor \times \left\lfloor \frac{(D_\text{S} - D_\theta + 1)}{D_\text{i}} \right\rfloor \times \left\lfloor \frac{(LD_\text{S} - LD_\theta + 1)}{LD_\text{i}} \right\rfloor$$  

(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.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_\theta$</td>
<td>$S_\theta$</td>
<td>$S_\text{i}$</td>
<td>111</td>
<td>111</td>
<td>111</td>
</tr>
<tr>
<td>$LS_\theta$</td>
<td>$LS_\theta$</td>
<td>$LS_\text{i}$</td>
<td>571</td>
<td>581</td>
<td>471</td>
</tr>
<tr>
<td>$D_\theta$</td>
<td>$D_\theta$</td>
<td>$D_\text{i}$</td>
<td>141</td>
<td>141</td>
<td>131</td>
</tr>
<tr>
<td>$LD_\theta$</td>
<td>$LD_\theta$</td>
<td>$LD_\text{i}$</td>
<td>331</td>
<td>341</td>
<td>331</td>
</tr>
<tr>
<td>je LD</td>
<td>12</td>
<td>16</td>
<td>12</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>insgesamt</td>
<td>12</td>
<td>32</td>
<td>12</td>
<td>27</td>
<td>36</td>
</tr>
</tbody>
</table>

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$^2$ verursacht. Bild 6.4 gibt die Anzahl richtig erkannter Verschiebungen in Abhängigkeit der Zufallszahlengenerator, der Intervallwerte aus Tabelle 24 sowie einem Startversatz (Offset) in 100 ms-Schritten und drei Spuren an.

Die hierbei verwendeten Zufallsgeneratoren

- `rand()`  
- L'Écuyer  
- subtraktive Methode  
- linear kongruente Methode  
  
Zufallsgenerator des XENIX-Betriebssystems [181],  
überlagerte Zufallsgeneratoren (16 und 32bit Version) [122],  
nach [116] und
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 Ereignisspurzeugung je Offset ca. 24 Stunden, woraus sich eine Rechendauer (180286) 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 unterstüzt 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 Laufzeitan teil 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.

<table>
<thead>
<tr>
<th></th>
<th>kurze Spur</th>
<th>lange Spur</th>
<th>Verhältnis (lang:kurz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datenmenge</td>
<td>15536 Ereignisse</td>
<td>98303 Ereignisse</td>
<td>6.327</td>
</tr>
<tr>
<td>Laufzeit</td>
<td>28.3 s</td>
<td>469.28 s</td>
<td>16.582</td>
</tr>
</tbody>
</table>

Tabelle 25: Laufzeit in Abhängigkeit von der Spurlänge
7 Einsatz des verteilten Meßsystems

Einige Beispiele zeigen anhand durchgeführter Messungen an unterschiedlichen Stellen der Objekt- 
systeme 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 
gesetzt. Die zeitliche Auflösung beträgt 1.28 µs. Die Objekt systeme 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-Quellcode mit den Meßstatements ①...⑥ bestückt:

- Beginn der Übergabe eines Auftragspuffers an das Transportsystem,
- Eintrag des Auftrags in IHI-interne Referenztablelle,
- Signal an Subsystem zur Übernahme des Auftrags (durch DMA),
- Ende der Auftragsübergabe bei Applikation,
- Applikation beginnt auf abgearbeiteten Auftrag zu warten,
- empfangener Rahmen wird über ein Unterbrechungssignal dem IHI-Treiber bekanntgemacht,
- Rückübersetzung der Adressen und Zuordnen des Pakets an die Applikation und
- 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 3 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(3) liegt.

<table>
<thead>
<tr>
<th>Bedeutung</th>
<th>Anzahl</th>
<th>Minimalwert</th>
<th>Mittelwert</th>
<th>Maximalwert</th>
<th>Varianz</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHI-Treiber intern</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 -&gt; 1</td>
<td>968</td>
<td>0.4390</td>
<td>0.5803</td>
<td>1.02656</td>
<td>4.7860E-10</td>
</tr>
<tr>
<td>5 -&gt; 3</td>
<td>968</td>
<td>1.6281</td>
<td>6.0728</td>
<td>22.8059</td>
<td>4.2108E-2</td>
</tr>
<tr>
<td>1 -&gt; 7</td>
<td>968</td>
<td>1.2979</td>
<td>1.3832</td>
<td>3.4270</td>
<td>2.6090E-8</td>
</tr>
<tr>
<td>T(3)</td>
<td>968</td>
<td>0.1950</td>
<td>0.2457</td>
<td>0.7475</td>
<td>2.9676E-9</td>
</tr>
<tr>
<td>Schreibaufräge (1 -&gt; 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T(ihi_write(OPEN))</td>
<td>4</td>
<td>4.6720</td>
<td>4.6963</td>
<td>4.7104</td>
<td>-</td>
</tr>
<tr>
<td>T(ihi_write(CR))</td>
<td>4</td>
<td>3.9641</td>
<td>4.0124</td>
<td>4.0921</td>
<td>-</td>
</tr>
<tr>
<td>T(ihi_write(VC))</td>
<td>957</td>
<td>3.8061</td>
<td>3.8824</td>
<td>7.8172</td>
<td>1.6427E-8</td>
</tr>
<tr>
<td>Leseaufräge (3 -&gt; 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T(ihi_read(OPEN))</td>
<td>4</td>
<td>3.3894</td>
<td>3.4192</td>
<td>3.4496</td>
<td>-</td>
</tr>
<tr>
<td>T(ihi_read(CR))</td>
<td>4</td>
<td>15.1398</td>
<td>1529.9470</td>
<td>6073.6060</td>
<td>-</td>
</tr>
<tr>
<td>T(ihi_read(VC))</td>
<td>957</td>
<td>10.4012</td>
<td>17.4746</td>
<td>1478.3050</td>
<td>4.302888E-3</td>
</tr>
</tbody>
</table>

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 (1->3) kaum, jedoch beim Empfang der Aufträge (3->5) 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 abfragt, 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.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Name</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PStart</td>
<td>Beginn eines Pollingschrittes (1)</td>
</tr>
<tr>
<td>1</td>
<td>PMOleer</td>
<td>Zugriff auf leeres (lokal abgebildetes) Objekt (2)</td>
</tr>
<tr>
<td>2</td>
<td>PMOfertig</td>
<td>Zugriffsende (4)</td>
</tr>
<tr>
<td>3</td>
<td>PMOlesen</td>
<td>Zugriff auf entferntes Objekt (2)</td>
</tr>
<tr>
<td>4</td>
<td>ihiwritevor</td>
<td>Aufruf zur Übergabe einer SDU an Schicht 4 (6)</td>
</tr>
<tr>
<td>5</td>
<td>ihiwriteach</td>
<td>Aufrufsende (4)</td>
</tr>
<tr>
<td>6</td>
<td>ihreadvor</td>
<td>Aufruf zum Lesen einer SDU von Schicht 4 (6)</td>
</tr>
<tr>
<td>7</td>
<td>ihreadnach</td>
<td>Aufrufende (6)</td>
</tr>
<tr>
<td>8</td>
<td>PMOAusw</td>
<td>Auswertung (2)</td>
</tr>
<tr>
<td>9</td>
<td>PStop</td>
<td>Ende eines Pollingschrittes (2)</td>
</tr>
<tr>
<td>10</td>
<td>PANachklapp</td>
<td>sofortiger Neubeginn eines Pollingschrittes (2)</td>
</tr>
<tr>
<td>11</td>
<td>PArvmsg</td>
<td>Warten auf neuen Pollingauftrag</td>
</tr>
</tbody>
</table>

Tabelle 27: Phasen beim Abfragen der Stationsobjekte (Managed Objects, MO)

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 7.4: Schrittweise Vergrößerung: a) ein Pollingschritt mit sofortigem Pollingneubeginn, b) Abfrage lokaler (leerer) und entfernter Objekte, c) Abfrage lokaler Objekte
7.1.3 TELNET-Applikation


Bild 7.6: Struktur der KA9Q Internet-Software unter XENIX (shm shared memory)


Bei der Multitasking-Implementierung werden nicht alle Daten aus den Netzrahmen unmittelbar an
<table>
<thead>
<tr>
<th>Bedeutung</th>
<th>Anzahl</th>
<th>Minimalwert</th>
<th>Mittelwert</th>
<th>Maximalwert</th>
<th>Var.Koeffizient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rahmen empfangen (ISO 2b)</td>
<td>269</td>
<td>0.01938</td>
<td>1.05336</td>
<td>18.8841</td>
<td>2.17200</td>
</tr>
<tr>
<td>Paket empfangen (IP)</td>
<td>268</td>
<td>0.04547</td>
<td>1.04691</td>
<td>18.8212</td>
<td>2.18739</td>
</tr>
<tr>
<td>Daten empfangen (Telnet)</td>
<td>152</td>
<td>0.16840</td>
<td>1.21774</td>
<td>11.8889</td>
<td>1.21591</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Senderichtung</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Daten eingelesen (Telnet)</td>
<td>196</td>
<td>0.05289</td>
<td>0.92989</td>
<td>11.8394</td>
<td>1.7697</td>
</tr>
<tr>
<td>Paket zu senden (TCP)</td>
<td>201</td>
<td>0.03288</td>
<td>0.91775</td>
<td>11.6262</td>
<td>1.7306</td>
</tr>
<tr>
<td>Rahmen senden (ISO 2b)</td>
<td>451</td>
<td>0.01908</td>
<td>0.62100</td>
<td>18.7271</td>
<td>2.9481</td>
</tr>
</tbody>
</table>

Tabelle 28: Meßpunktabstände bei Telnet-Anwendung für ein Meßintervall von 4:43.315 min (gemessene Zeiten in ms)


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 kleineren Blockes erkannt, woraufhin er die Verbindung abbaut. SEND holt während der Transferphase einen Datenblock vom Hintergrundspeicher (1, 2) und übergibt diesen an das Trans-
portsystem (9, 10). Sind nur noch smin Sendepuffer frei, so wird geprüft, ob ein Sendepuffer zurückgelesen werden kann (9, 11). Erst wenn alle smax Sendepuffer belegt sind, wird auf die Rücknahme eines Sendepuffers vom Transportsystem gewartet (7, 8).

Im Empfangsprozeß wird ein Wechselpuffermechanismus verwendet. Zuerst wird bei (1, 2) ein Empfangspuffer an die Schicht 4 übergeben. Nun wird in (3, 4) auf die Rücknahme eines Puffers gewartet. Bevor dieser Puffer auf dem Hintergrundspeicher abgelegt wird (7, 8), wird der andere Puffer an das Transportsystem übergeben (5, 6). 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 (1 2 3 4 5 6 7 8 → 1...).

Die Synchronität zwischen Sender und Empfänger erlaubt, daß im Statistikinterpreter die Musterpaaare der lokalen Prozesse und das Paar SEND③-REC④ korrekt zugeordnet werden können, während die Zuordnung bei SEND①-REC⑥ 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 7.8: Synchroner Dateitransfer zwischen Sender (oben) und Empfänger (unten) (SEND → REC graue Markierungen bei n=1,2,...)

<table>
<thead>
<tr>
<th>Bedeutung</th>
<th>Anzahl</th>
<th>Minimalwert</th>
<th>Mittelwert</th>
<th>Maximalwert</th>
<th>Varianz</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 -&gt; 2</td>
<td>257</td>
<td>0.6528</td>
<td>5.4581</td>
<td>88.7475</td>
<td>7.6004E-5</td>
</tr>
<tr>
<td>2 -&gt; 1</td>
<td>257</td>
<td>2.6649</td>
<td>2.7724</td>
<td>3.8118</td>
<td>4.2355E-8</td>
</tr>
<tr>
<td>3 -&gt; 1</td>
<td>259</td>
<td>17.3081</td>
<td>158.7004</td>
<td>429.0637</td>
<td>2.3813E-3</td>
</tr>
<tr>
<td>RECEIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 -&gt; 1</td>
<td>256</td>
<td>27.6582</td>
<td>151.1372</td>
<td>218.7904</td>
<td>1.7259E-3</td>
</tr>
<tr>
<td>1 -&gt; 2</td>
<td>256</td>
<td>0.0563</td>
<td>0.0601</td>
<td>0.18176</td>
<td>1.1740E-10</td>
</tr>
<tr>
<td>1 -&gt; 3</td>
<td>257</td>
<td>0.5120</td>
<td>14.0300</td>
<td>48.1715</td>
<td>9.7380E-6</td>
</tr>
<tr>
<td>SEND-RECEIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEND1 -&gt; REC1</td>
<td>250</td>
<td>0.19456</td>
<td>159.9341</td>
<td>434.8160</td>
<td>2.9697E-3</td>
</tr>
<tr>
<td>SEND2 -&gt; REC1</td>
<td>254</td>
<td>10.9030</td>
<td>17.3211</td>
<td>76.2368</td>
<td>1.7791E-5</td>
</tr>
<tr>
<td>Abstand (SEND1)</td>
<td>256</td>
<td>31.5238</td>
<td>166.9127</td>
<td>477.5014</td>
<td>2.2634E-3</td>
</tr>
<tr>
<td>Abstand (REC1)</td>
<td>256</td>
<td>39.0016</td>
<td>165.7790</td>
<td>230.7072</td>
<td>1.6773E-3</td>
</tr>
</tbody>
</table>

Tabelle 29: Meßwerte für synchronen Dateitransfer (gemessene Zeiten in ms)
8 Zusammenfassung und Schlußbemerkungen


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 Entkoppelung über das FIFO-Element. Damit wird aber der Abfragebus breiter:

- Signalleitungen zum Takten und Rücksetzen des Zählers in den Meßfühlern
- 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 differenzielle Leitungspaaare 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 gesenderter bzw. empfangener Rahmen bedeutet dies, daß wie bei der SYNC-Mustererkennung das Rahmenende zeitlich markiert und bis zur endgültigen Entscheidung zwischen-gespeichert 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 spezifizier-

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 Ereignisin-
trags gewertet werden. Die meisten Änderungen erfolgten im Statistikinteroper als Anbindung 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. Reproduzierba-
re Ergebnisse mit zusätzlicher Systemlast verlangen, daß die auf den einzelnen Objektsystemen ablaufenden Programme kontrolliert angestoßen und während ihrer Laufzeit überwacht werden kön-
fehlszeile inklusive aller Parameter erhalten und den Programmzustand an die steuernde Station meldem. 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 je-
doch oft, daß für diese Werte eine Approximation durch eine bekannte Verteilungsfunktion durchgeführ 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:
Bild 8.1: Prozentuale Netzlast beim Einsatz des verteilen 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].


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 zeitaufwändige 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: 'vart' '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


[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


[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


[95] INTEL CORPORATION: System 310AP, Intel Corporation


Part 4: Management Framework, 1989


Annex A: Acknowledged Connectionless Service, October 1986


120] K. Kümmerle: Höhere Programmiersprachen und ihre Compiler, Vorlesungsskript, Universität Stuttgart


122] P. L'Ecuyer: Efficient and Portable Combined Random Number Generators, *Communications of the ACM*, vol. 31, no. 6, June 1888, pp. 742-774


[147] MOTOROLA: MC68824 Token Bus Controller, Motorola Inc., Datenblatt 1986


[171] TH.J.ROUTT: From TOP (3.0) to bottom: Architectural close-up, *Data Communications*, May 1988, pp.237-256


GUD: Installationshinweise, GUD Datensysteme GmbH, November 1989

[186] SIEMENS: Schnittstellenbeschreibung zur Firmware auf der CP536*, Siemens AG, 27. April 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


[191] P.R. STRAUSS: Flexibility or heresy? The struggle to redraw MAP, *Data Communications*, November 1988, pp.49-56


C. HEDRICK: Telnet Terminal Speed Option, *RFC 1079*, December 1988
J. VANBOKKELEN: Telnet Terminal-Type Option, *RFC 1091*, February 1989
R. BRADEN: Requirements for Internet Hosts - Application and Support, *RFC 1123*, October 1989


