MONITORING VON TRANSPUTER-NETZWERKEN

Dipl. Inform. Torsten Braun, Dipl. Inform. Martina Zitterbart
Universität Karlsruhe, Institut für Telematik, Karlsruhe.

1 Zusammenfassung

Bei der Implementierung von Software auf Multiprozessorsystemen, z.B. Transputer-Netzwerken, stellt sich allgemein das Problem, daß ein Mangel an hierfür geeigneten Debugging-Tools und Werkzeugen für eine detaillierte Leistungsuntersuchung der Software auf den Transputer-Netzwerken herrscht.

Dieser Beitrag diskutiert grundlegende Probleme beim Monitoring von Multiprozessorsystemen, speziell von Transputer-Netzwerken, und stellt geeignete Lösungsmöglichkeiten vor.

Hinsichtlich verschiedener Kriterien wie Genauigkeit, Auflösung, Overhead, Hardware- und Instrumentierungsaufwand usw. werden verschiedene Monitore miteinander verglichen und ihre Anwendungsbereiche dargestellt.

2 Monitore

Monitore werden eingesetzt, um charakteristische Leistungsdaten zu messen und das dynamische Verhalten eines Systems zu beobachten. Leistungsmessungen geben Aufschluß über die Leistungsfähigkeit des Rechners und der darauf rechnenden Software, während durch die Beobachtung des Systemverhaltens Fehler oder Schwachstellen, die zu einer geringeren Leistung führen, aufgedeckt werden können.

Ein Monitor setzt sich nach /1/ aus Sensor, Transformator und Anzeige zusammen (Bild 1). Der Sensor nimmt Daten auf, die durch den Transformator in eine geeignete Form zur Weiterverarbeitung umgewandelt werden. Die Ergebnisse werden von einer Anzeige-Einheit dargestellt.

Bild 1 Aufbau eines Monitors

Grundsätzlich werden Monitore in Software-, Hardware- und Hybridmonitore eingeteilt.

Ein **Hardware-Monitor** wird mit Hilfe von Sensoren an das zu beobachtende System angeschlossen, erfaßt elektrische Signale und wertet diese dann aus. Bei dieser Methode wird das System zwar nicht belastet und Auflösung und Genauigkeit sind sehr gering, doch gestaltet sich die Zuordnung von Signalen zu Ereignissen schwierig.

Ein **Software-Monitor** ist ein spezielles Programm, welches in das zu vermessende System eingegliedert ist. Er sammelt Daten, die von einem oder mehreren Meßprogrammen erzeugt werden, welche zusammen mit der zu vermessenden Software rechnen. Deshalb ist es zwar nicht sehr schwierig, einen Zusammenhang zwischen den gesammelten Daten und den Ereignissen herzustellen, jedoch erzeugen die Meß- und Auswerteprogramme einen nicht unerheblichen Overhead (Ausführungszeit der Programme, Speicherbedarf). In der Regel sind Auflösung und Genauigkeit schlechter als bei Hardware-Monitoren. Oft sind Software-Monitore nur mit erheblichem Aufwand auf andere Systeme portierbar. Allerdings erfordern sie keinen zusätzlichen Hardware-Aufwand.

Ziel von **Hybrid-Monitoren** ist es, die jeweiligen Vorteile von Software- und Hardware-Monitor zu nutzen und deren Nachteile zu vermeiden. Die Software erzeugt Ereignisse, die durch Hardware-Komponenten erfaßt werden. Beispiele für Hybrid-Monitore sind die verteilten Monitorsysteme NETMON-II /2/ und Zählmonitor /3/.

Hinsichtlich der Meßwerterfassung wird zwischen ereignis- und zeitgesteuerten Monitoren unterschieden /3/. Bei **zeitgesteuerten** Monitoren wird in gewissen Zeitabständen der Zustand des Systems aufgezeichnet, bei **ereignisgesteuerten** Monitoren veranlaßt ein bestimmtes Ereignis das Aufzeichnen einer Kennung und der Zeit. Diese beiden Informationen charakterisieren das Ereignis und bilden zusammen einen Ereignis-Rekord. Das ereignisgesteuerte Monitoring basiert auf der Auswertung von Ereignispaaren. Ein Ereignispaar besteht aus zwei Ereignis-Rekords und beschreibt den Anfang und das Ende eines Vorgangs, so daß sich dessen Dauer quantitativ ermitteln läßt.

3 Probleme beim Monitoring von Multiprozessorsystemen

Multiprozessorsysteme lassen sich nur dann mit einem einzigen Monitor beobachten, wenn die einzelnen Prozessoren nicht räumlich verteilt sind und wenn der Monitor leistungsfähig genug ist, eine größere Anzahl von Prozessoren gleichzeitig zu beobachten.

Ansonsten erfordert die Beobachtung von Multiprozessorsystemen bzw. deren Leistungsmessung verteilte Monitorsysteme. Ein Beispiel für ein solches verteiltes Monitorsystem ist NETMON-II. Es besteht aus einer zentralen Kontrollstation, die den globalen Ablauf koordiniert und aus mehreren Monitorstationen, die jeweils ein Meßobjekt beobachten (s. Abschnitt 4.3).

Das Hauptproblem beim Monitoring von Multiprozessorsystemen mit verteilten Monitorsystemen besteht darin, aus den lokalen Beobachtungen der einzelnen Meßstationen globale Ergebnisse abzuleiten. Bei einer ereignisgesteuerten Messung fallen durch die einzelnen Monitorstationen lokale Ergebnisse in Form von voneinander unabhängigen Ereignisfolgen an. Um eine Beziehung zwischen den lokalen Ergebnissen herstellen zu können, müssen die Ereignisfolgen einer globalen Zeit zugeordnet werden. Hierzu gib es zwei Ansätze:

- Die Uhren der verteilten Monitorsysteme werden zyklisch synchronisiert. Dies kann durch Hardware (NETMON-II) oder Software (Software-Monitor in Abschnitt 4.1) geschehen. Für Transputer-Netzwerke kann mit einem leicht abgewandelten Algorithmus aus /4/ für eine zyklische Synchronisation der Uhren erfolgen.

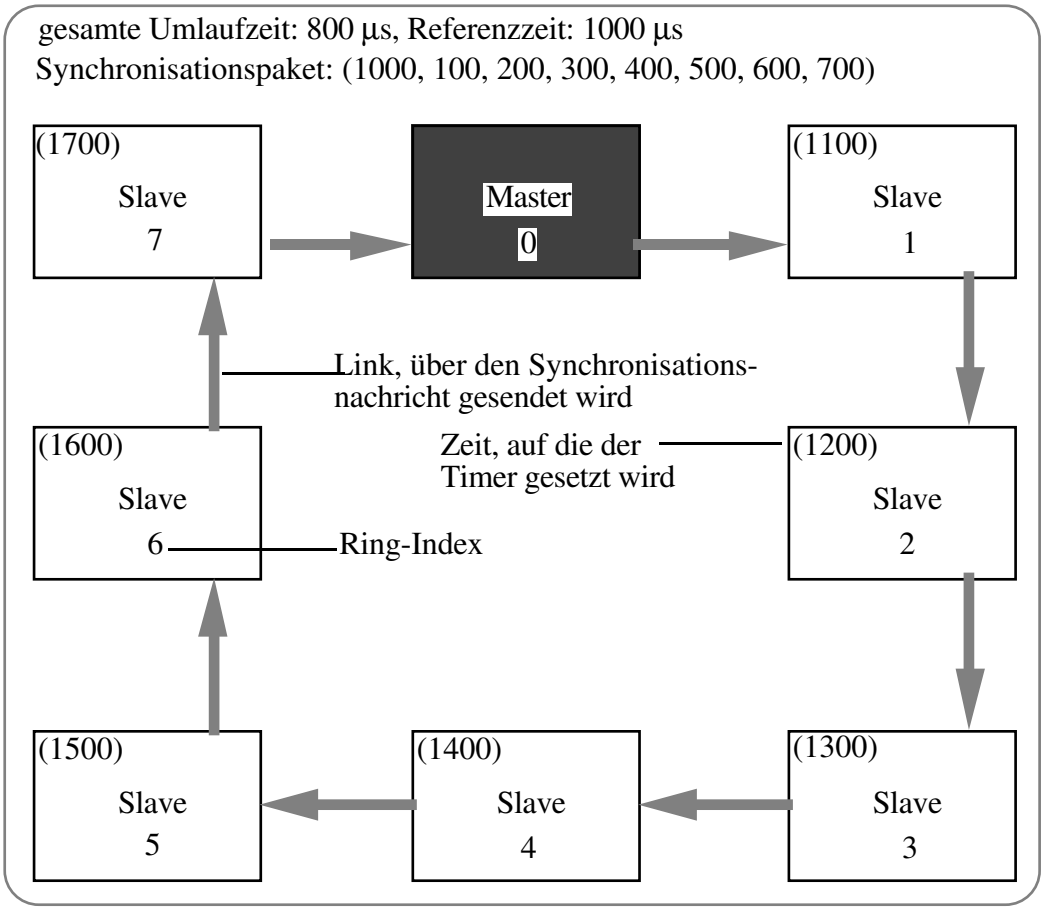
  Der Algorithmus erfordert eine Ring-Struktur des Transputer-Netzes. Der Ablauf geschieht in drei Phasen: der Zähl-, der Simulations- und der Synchronisationsphase.

  In einer Zählphase werden die Transputer gezählt und jedem Transputer ein Ring-Index (1..n) zugewiesen.

  In der Simulations- und der Synchronisationsphase werden jeweils identische Operationen ausgeführt. Dabei sendet ein ausgezeichneter Transputer (Master) über den Ring an die nachfolgenden Transputer (Slaves) eine Synchronisationsnachricht, welche die aktuelle Zeit des Masters (Referenzzeit) und die individuelle erwartete Verzögerung für das Senden der Synchronisationsnachricht vom Master zum jeweiligen Slave enthält. Die Zeit, auf die ein Slave-Timer nun gesetzt werden muß, berechnet sich aus der Summe der Referenzzeit und der erwarteten Verzögerung für diesen Slave. Die Nachricht wird dann von jedem Slave empfangen und an den jeweiligen Nachfolger weitergesendet bis schließlich der Master-Transputer die Nachricht wieder zurückerhält.

  Die jeweilige erwartete Verzögerung der Nachricht zwischen Master und Slave berechnet der Master aus der benötigten Umlaufzeit der Nachricht aus der Simulationsphase, indem die gesamte Umlaufzeit der Synchronisationsnachricht durch die Anzahl der Transputer dividiert und mit dem jeweiligen Slave-Index multipliziert wird. Die einzelnen Verzögerungszeiten werden dann an den Slaves durch ihren Ring-Index bekannten Positionen der Synchronisationsnachricht eingetragen.

Im Bild 2 ist der Algorithmus anhand eines Beispiels erklärt. Es werden 8 Transputer synchronisiert. Die Umlaufzeit beträgt 800 µs, d.h. das Weitergeben der Synchronisationsnachricht zwischen zwei Transputern beträgt 100 µs. Die Synchronisationsnachricht enthält die Referenzzeit (1000) und die jeweiligen Verzögerungen zwischen Master und Slave i (i = 1…7). Die einzelnen Timer werden dann diesen Werten entsprechend gesetzt.



Die Synchronisationsphase kann mehrmals durchgeführt werden. Die Häufigkeit, wie oft die Transputer synchronisiert werden müssen, hängt vom Drift ihrer Timer ab.

- Für die anfallenden Daten kann auch erst nach einer Messung eine gemeinsame Zeitbasis oder zumindest eine zeitliche Ordnung hergestellt werden.

  In /5/ wird ein Verfahren vorgestellt, mit dem sich nach dem Programmablauf durch Analyse der lokalen Ereignisfolgen eine globale Zeit abschätzen läßt. Es wird angenommen, daß zwei lokale Uhren durch Kristall-Oszillatoren gesteuert werden und daß eine Uhr (bei System A) als perfekt betrachtet werden kann. Die Abweichung der zweiten Uhr (System B) beträgt dann

  $$\Delta(t) = a(t_0) + b * (t - t_0),$$

  ($a(t_0)$: Initialisierungsfehler, b: Frequenz-Offset).

  Nach dem Programmablauf werden die durch das Senden und Empfangen einer Nachricht erzeugten Ereignispaare analysiert. Das Ziel ist, die beiden Parameter a und b in der obigen Gleichung möglichst genau zu bestimmen. In der Regel sind die Kommunikationsverzögerungen nicht konstant und unterliegen einer unbekannten Wahrscheinlichkeitsverteilung. Zum Abschätzen der beiden Parameter werden verschiedene Methoden, wie z.B. Regressionsanalyse, vorgeschlagen.

Weitere Probleme beim Design von Monitoren für verteilte Systeme oder Multiprozessorsysteme sind z.B. das Speichern, die Analyse und die Bewertung von Meßdaten sowie die geeignete Darstellung von Ergebissen /6/, /7/.

4 Monitore für Transputer-Netzwerke

Am Institut für Telematik der Universität Karlsruhe wurden mehrere Monitortypen für Transputer-Netze entwickelt und im praktischen Einsatz erprobt. Im einzelnen waren dies neben einem zeitgesteuerten Software-Monitor ("Profiler") drei ereignisgesteuerte Monitore: ein Software-Monitor, ein Logikanalysator und das verteilte Monitorsystem NETMON-II.

Die ereignisgesteuerten Monitore arbeiten alle nach dem gleichen Prinzip /8/. Der Benutzer muß Anweisungen, die Ereignisse erzeugen, in die zu untersuchende Software einfügen. Die Ereignisse, die während des Programmablaufs auftreten, werden von den verschiedenen Monitoren aufgezeichnet. Danach kann der Benutzer die Ergebnisse, die vom Monitor in Form von Ereignislisten geliefert werden, mit Hilfe von unterstützenden Programmen auswerten.

Die Ereignisse werden bei Transputer-Software so plaziert, daß sich aus ihrer Reihenfolge in der Ereignisliste und den dazugehörigen Zeiten nachvollziehen läßt, wann und wie lange ein Prozeß den Prozessor belegt bzw. aktiv ist, und warum auf dem Prozessor eventuell kein Prozeß rechnet. Dies ist möglich, weil Prozesse auf Transputern nur nach bestimmten Anweisungen deaktiviert werden können, z.B. Kommunikationsoperationen, explizite Deaktivierung für eine bestimmte Zeit oder alternatives Warten.

Werden nun z.B. direkt vor und hinter einer Kommunikationsoperation Ereignisse erzeugt, so wird in der Regel kurz nach dem ersten Ereignis der Prozeß deaktiviert, die Zeit des zweiten Ereignisses gibt den Zeitpunkt an, an dem der Prozeß wieder in den Zustand rechnend übergeht.

Diese Monitore werden im folgenden kurz beschrieben und bewertet.

4.1 Software-Monitor

Beim ereignisgesteuerten Software-Monitor wird ausgenutzt, daß jeder Transputer einen eigenen internen Systemtimer besitzt. Dieser Timer hat für Prozesse mit niedriger Priorität eine Auflösung von 64 µs, für Prozesse hoher Priorität eine Auflösung von 1 µs.

In jedem Transputer wird durch eine Initialisierungsprozedur ein Speicherbereich gewünschter Größe reserviert, in dem die Ereignis-Rekords gespeichert werden können.

Beim Software-Monitor müssen folgende Probleme gelöst werden:

- Die Meßergebnisse (Ereignislisten) werden jeweils auf dem beobachteten Transputer abgespeichert, die Auswertung erfordert aber sämtliche Meßergebnisse in einem einzigen Speicherbereich. Die einzelnen Ereignislisten müssen deshalb von einem Transputer nach einer Messung eingesammelt und die Ereignisse nach ihrem zeitlichen Auftreten geordnet werden. Dazu wird ein aus dem Transputer-Netz ein Baum mit möglichst geringer Tiefe abgeleitet, der alle Transputer des Netzes genau einmal enthalten muß. Die einzelnen Knoten des Baums sammeln jeweils die Ergebnisse der untergeordnten Knoten ein. Zum Einsammeln werden die Ergebnisse über Links gesendet. Schließlich erhält ein Transputer die gesamten Meßergebnisse.
- Die abgespeicherten Zeiten in den einzelnen Ereignislisten sind nicht global, sondern nur lokal gültig. Dies liegt am Drift bei unterschiedlicher Taktversorgung und an der unterschiedlichen Initialisierung. Die Initialisierung der Timer geschieht beim Rücksetzen der Transputer, das immer dann erfolgt, wenn ein Programm geladen wird. Wird ein ganzes Transputer-Netzwerk geladen, so werden die jeweiligen Transputer zu unterschiedlichen Zeitpunkten geladen, so daß die Werte der Timer zu einem Zeitpunkt nicht übereinstimmen.

  Um global gültige Zeiten zu erhalten, wurden die beiden in Abschnitt 3 beschriebenen Methoden verwendet. Mit dem dort beschriebenen Synchronisationsalgorithmus werden die einzelnen Timer zu Beginn einer Messung synchronisiert. Während der Messung wurde auf den Einsatz

dieses Algorithmus verzichtet, um den Overhead möglichst gering zu halten. Eventuelle, z.B. durch Drifteffekte verursachte Abweichungen der Timer von der globalen Zeit werden nach der Messung korrigiert. Der Drift zwischen unterschiedlichen Taktgebern ist allerdings sehr gering (ca. 10^{-5}). Die Korrektur läßt sich wegen der synchronen Kommunikation zwischen den Transputern relativ leicht durchführen.

Bei einer Kommunikation (Bild 3) synchronisieren sich die kommunizierenden Prozesse miteinander, d.h. der Prozeß der zuerst zur Kommunikation bereit ist, wartet auf den Partnerprozeß. Unmittelbar nach Beendigung der Kommunikation können die beiden Prozesse weiterrechnen. In diesem Fall sind die beiden Zeitpunkte t_1 und t_2 gleich, d.h. eventuelle Unterschiede lassen sich auf nicht synchrone Timer zurückführen.

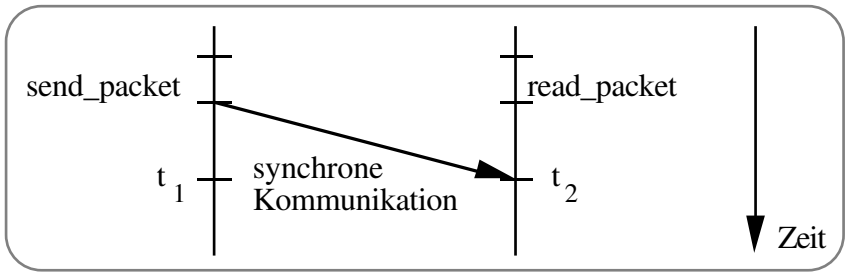


Bild 3 Synchrone Kommunikation zwischen Transputern

Beim Software-Monitor ist zusätzlicher Hardware-Aufwand nicht notwendig und es läßt sich eine beliebige Anzahl von Transputern mit ihm beobachten.

Nachteilig ist der relativ große Overhead durch das Abspeichern der Ereignis-Rekords (16 µs für die entsprechende Anweisung) während der Messung und zu Debugging-Zwecken ist der Software-Monitor nicht geeignet. Der Instrumentierungsaufwand der zu untersuchenden Software ist verhältnismäßig hoch, weil zusätzlich zum Einfügen von Anweisungen, die Ereignisse erzeugen, vom Anwender Anweisungen eingefügt werden müssen, welche die Synchronisation der verschiedenen Timer und die Zusammenführung der lokal anfallenden Ereignislisten unterstützen.

4.2 Logikanalysator

Als Hybrid-Monitor wurde ein Logikanalysator der Firma Rhode&Schwarz eingesetzt. Mit diesem Logikanalysator ist es aufgrund der Anzahl der vorhandenen Takteingänge möglich, bis zu vier Transputer gleichzeitig zu beobachten. Da der Logikananlysator kein verteiltes Monitorsystem darstellt, entfällt hier das Synchronisationsproblem.

Der Logikanalysator muß so eingestellt werden, daß er bestimmte Anweisungen auf dem Transputer, die ein Ereignis darstellen sollen, erkennt und das Ereignis mit der entsprechenden Zeit abspeichert. Ein Ereignis wird dadurch erzeugt, daß lesend oder schreibend auf eine Speicherstelle zugegriffen wird, deren Adresse auf der verwendeten Transputer-Karte nicht benötigt wird. Der Kennungswert ergibt sich aus dem Wert der Adresse. Der adressierbare Bereich beträgt beim Transputer 4 Gigabyte, meist werden aber nur wenige Megabyte pro Transputer benutzt, so daß die Anzahl der nicht benutzten Adressen für die Darstellung der Ereigniskennungen ausreicht.

Im Bild 4 ist der Anschluß von vier Transputern an den Logikanalysator dargestellt. Als Takteingang dient das Signal notMemS0, das mit der fallenden Flanke die Gültigkeit einer Adresse anzeigt. Der Logikananlysator wird also so konfiguriert, daß er bei der fallenden Flanke des Signals notMemS0 den Adreßwert überprüft und abspeichert, falls die Adresse eine Ereigniskennung repräsentiert.

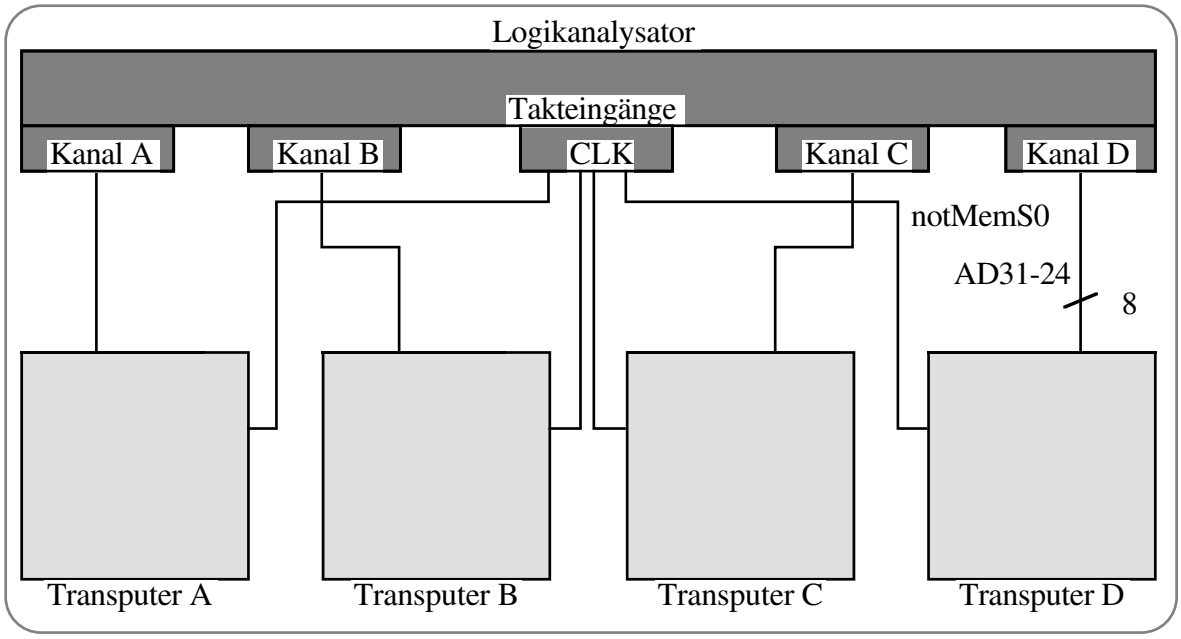


Bild 4 Anschluß von Transputern an einen Logikanalysator

Der wesentliche Vorteil des Logikanalysators liegt in der nicht notwendigen Synchronisation von Zeitgebern. Allerdings kann mit einem Logikanalysator nur eine bestimmte Anzahl von Prozessoren beobachtet werden, die von der Ausstattung und Leistungsfähigkeit des Logikanalysators abhängt.

Der durch die zusätzlichen Anweisungen entstehende Overhead beträgt typischerweise nur 1-2% und ist im Vergleich zum Software-Monitor (bis zu 100%) sehr gering. Die Auflösung von 100 ns liegt unter der Speicher- bzw. Befehlszykluszeit des Transputers und ist deshalb ausreichend. Die Auswertemöglichkeiten des Logikanalysators selbst sind beschränkt, mit etwas Aufwand läßt sich dieser Nachteil allerdings beheben, da der Logikanalysator Schnittstellen zum Datenaustausch mit PCs besitzt, so daß Meßdaten auf einen PC übertragen und dort mit leistungsfähigeren Programmen ausgewertet werden können. Etwas Probleme bereitet die Tatsache, daß schwer zugängliche Signale direkt vom Transputer abgegriffen werden müssen, was in der Regel Änderungen der Hardware erfordert.

4.3 NETMON-II

Auch NETMON-II kann als Hybrid-Monitor eingestuft werden. NETMON-II wurde am Institut für Telematik als Monitorwerkzeug für verteilte Systeme entwickelt und für den Einsatz bei Transputer-Netzen modifiziert /2/, /9/.

NETMON-II besteht aus verteilten Meßstationen (Monitorstationen) und einer zentralen Kontrollstation, die durch einen Token Ring miteinander verbunden sind (Bild 5). Die Kommunikation wird durch ein OSI-Transportsystem unterstützt.

Die Kontrollstation benötigt neben der Kommunikationssoftware und dem Mehrprozeß-Kernel Software für die Systemkontrolle, die On-line-Auswertung und den File-Server-Betrieb. Die Off-line-Auswertung ist auf VAX-Rechnern vorgesehen.

Für jedes Meßobjekt existiert eine Monitorstation, welche über einen gemeinsamen Zeitkanal synchronisiert werden. Die Messung erfolgt ereignisgesteuert, wobei die Monitorstationen die von den Meßobjekten ausgegebenen Ereigniskennungen aufzeichnen, die dann von den Monitorstationen und der Kontrollstation on-line oder off-line verarbeitet werden. Ereigniskennungen können mit Hilfe eines Ereignis-Filters gefiltert werden, um bereits frühzeitig eine Reduzierung der anfallenden Datenmengen zu erreichen.

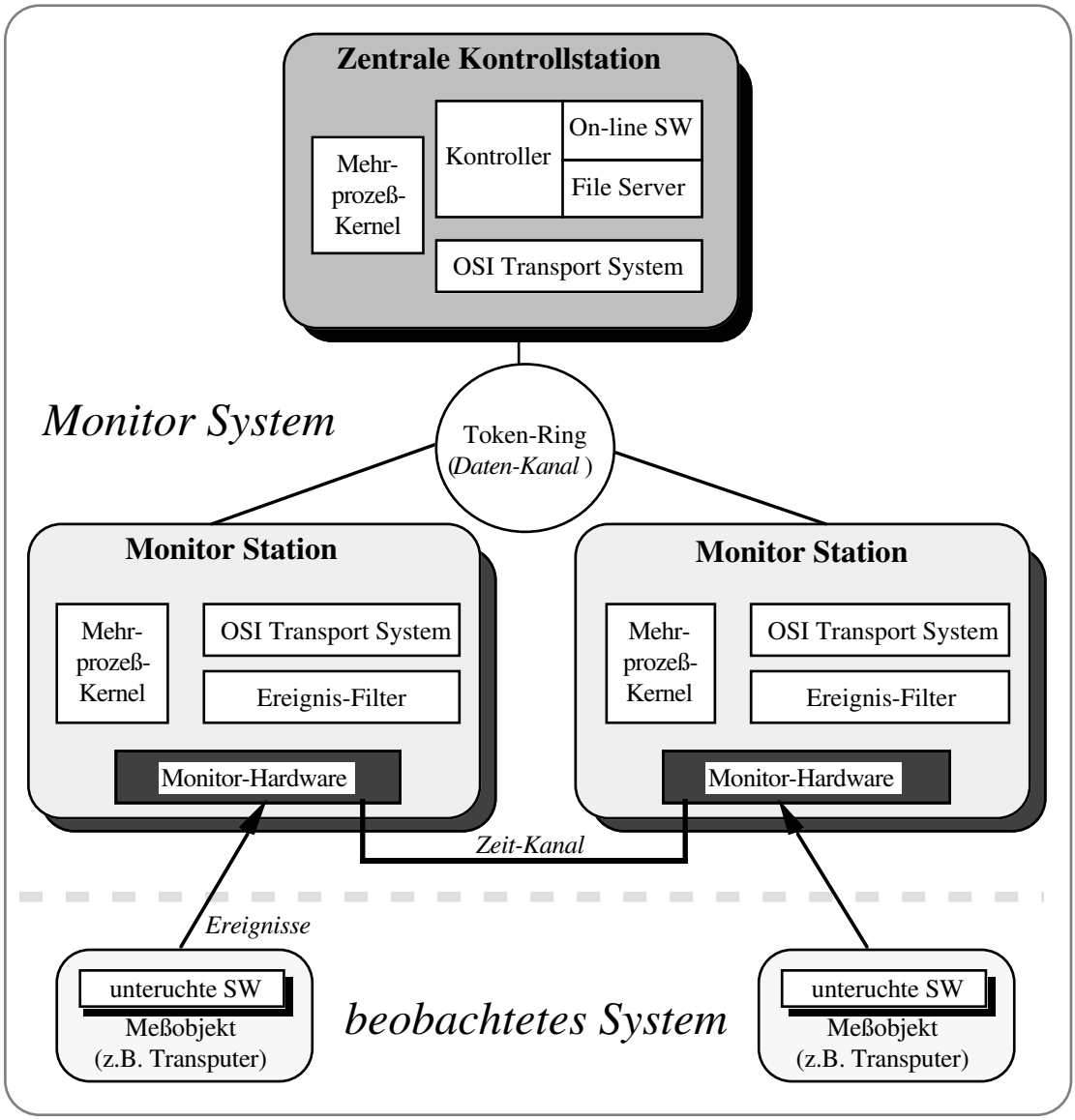


Bild 5 NETMON-II

Die Monitor-Hardware der Monitorstationen hat die Aufgabe, die Ereignisse zu erfassen und sie mit Zeitstempeln zu versehen. Der Anschluß eines Meßobjekts an die Monitor-Hardware erfolgt über eine Centronics-ähnliche Schnittstelle. Für Transputer wurden deshalb zwei Interfaces entwickelt, die Daten aufnehmen und an die Monitorstationen byteweise übergeben.

Beim ersten Interface werden Daten (= Ereigniskennungen) über einen Transputer-Link ausgegeben, das Interface parallelisiert diese und übergibt sie gemäß einem speziellen Protokoll den Meßstationen /10/. Die Ereigniskennungen haben eine Länge von drei Bytes. Diese Methode hat allerdings den Nachteil, daß Ausgaben über Links relativ langsam sind und daß durch die Link-Ausgabe der rechnende Prozeß deaktiviert und damit der gesamte Ablauf auf einem Transputer wesentlich beeinflußt wird.

Bei einem verbesserten Verfahren erfolgt die Ausgabe einer Ereigniskennung über memory-mapped E/A, d.h. Register des Interfaces werden direkt an die Speicherschnittstelle des Transputers angeschlossen. Das Interface reagiert beim schreibenden Zugriff, indem die geschriebenen Daten an die Meßstationen gemäß dem vereinbarten Übergabeprotokoll weitergereicht werden.

Der größte Vorteil von NETMON-II liegt in der Verfügbarkeit von leistungsfähigen Auswertemöglichkeiten. Als einziger hier betrachteter Monitor erlaubt NETMON-II die Auswertung während der Messung (on-line-Auswertung). Auch die Auflösung von 1 µs /11/ ist ausreichend, der Overhead ist wie beim Logikanalysator sehr gering. Für eine Anweisung, die ein Ereignis erzeugt, werden hier nur ca. 2,5 µs benötigt.

Etwas hinderlich ist der beträchtliche Hardware-Aufwand. Für jeden zu überwachenden Transputer ist derzeit eine komplette Monitorstation (ein Standard PC-AT mit Monitor-Hardware auf einer Einsteckkarte) sowie ein Transputer-NETMON-Interface notwendig. Es ist aber prinzipiell möglich, mehrere Einsteckkarten in einem PC zu betreiben. Weiter ist das Abgreifen der benötigten Transputer-Signale mit nicht unerheblichem Aufwand verbunden. Rein theoretisch sind beliebig viele Prozessoren beobachtbar.

4.4 Profiler

Bei den bisher beschriebenen Monitoren muß der Benutzer an bestimmten Stellen der Software Anweisungen einfügen, die ein Ereignis darstellen. Das Einfügen von Anweisungen wird beim sogenannten Profiler /12/ vermieden. Beim Profiler handelt es sich um einen Prozeß, der mit hoher Priorität auf einem Transputer läuft, zyklisch die niederprioren Prozesse unterbricht und dabei mit Hilfe des Workspace-Pointers die Identität des unterbrochenen Prozesses feststellt. Anschließend wird die Anzahl der erfolgten Unterbrechungen eines Prozesses pro Zeiteinheit ermittelt und daraus dessen Rechenzeit geschätzt. Je häufiger der Profiler die niederprioren Prozesse unterbricht, um so genauer sind die Ergebnisse, jedoch steigt dann der Overhead beträchtlich an. Mit dem Profiler lassen sich die Rechenzeiten relativ leicht, d.h. ohne großen Instrumentierungsaufwand ermitteln, genauere Aussagen über das Verhalten der Software auf den Transputern, insbesondere das Ermitteln von Wechselwirkungen zwischen verschiedenen Transputern sind mit dem Profiler aber kaum möglich, so daß dann wieder auf die ereignisgesteuerten Monitore zurückgegriffen werden muß.

5 Einsatzmöglichkeiten der Monitore

Die beschriebenen Monitore wurden zur Fehlersuche und zur Leistungsbewertung im Rahmen von Implementierungen von Kommunikationsprotokollen auf Transputernetzen eingesetzt /8/, /13/.

Bei der Fehlersuche bei Transputer-Netzen stellt sich oft das Problem, daß nur ein einziger Transputer Bildschirmausgaben tätigen kann, weil in der Regel nur ein Transputer mit dem Host-Rechner direkt verbunden ist. Um die anderen Transputer im Netzwerk beobachten zu können, sind Monitore wie der Logikanalysator oder NETMON-II unverzichtbare Hilfsmittel.

Leistungsbewertung hat zwei Ziele. Zum einen soll die Leistung eines Systems zu Vergleichszwecken bewertet werden, zum andern sollen damit mögliche Schwachstellen aufgedeckt werden können. Bei der Implementierung von Kommunikationsprotokollen konnten durch den Einsatz der beschriebenen Monitore Leistungssteigerungen von bis zu 50% /8/ erzielt werden, indem Kommunikations- und Prozessoruntätigkeitszeiten drastisch reduziert werden konnten.

Die beiden Software-Monitore erzeugen zwar einen sehr großen Overhead, sind aber zur Bestimmung von Prozeßrechenzeiten oder von Durchsatzergebnissen gut einsetzbar. Sie haben den nicht zu unterschätzenden Vorteil, daß kein zusätzlicher Hardware-Aufwand notwendig ist. Für genauere Untersuchungen wie zum Beispiel das Ermitteln von Kommunikations- oder Prozessoruntätigkeitszeiten muß auf die leistungsfähigeren Monitore Logikanalysator und NETMON-II ausgewichen werden.

6 Literatur

/1/ D. Ferrari: "Computer Systems Performance Evaluation", Prentice Hall 1978.

/2/ O. Endriss, M. Steinbrunn, M. Zitterbart: "NETMON-II a monitoring tool for distributed and multiprocessor systems", 4th International Conference on Data Communications and their Performance, Juni 1990, Barcelona, Spanien.

/3/ R. Hofmann, R. Klar, N. Luttenberger, B. Mohr, G. Werner: "An Approach to Montoring and Modeling of Multiprocessor and Multicomputer Systems", Proceedings of IFIP W.G. 7.3 International Seminar on Performance of Distributed and Parallel Systems, Kyoto, Japan, Dezember 1988.

/4/ U. de Carlini, U. Villano: "A Simple Algorithm for Clock Synchronization in Tranputer Networks", Software - Practice and Experience, Vol. 18, 4/1988.

/5/ A. Duda, G. Harrus, Y. Haddad, G. Bernard: "Estimating Global Time In Distributed Systems", Proceedings of 7th International Conference on Distributed Computing Systems, Berlin 1987.

/6/ M. Abrams: "Design of a Measurement for Distributed Systems", ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, 1988, Sante Fe, USA.

/7/ U. Herzog: "Performance evaluation principles for vector- and multiprocessor systems", Parallel Computing 7/1988.

/8/ T. Braun: "Monitoring von Transputer-Netzwerken am Beispiel des OSI-Internetzwerkprotokolls", Diplomarbeit am Institut für Telematik, Universität Karlsruhe, Januar 1990.

/9/ M. Zitterbart: "Monitoring and Debugging Transputer-Networks with NETMON-II", CONPAR´90, September 1990, Zürich, Schweiz.

/10/ T. Braun, C. Jost: "Lastgenerator und NETMON-Interface für Transputer-Netzwerke", Studienarbeit am Institut für Telematik, Universität Karlsruhe, Juni 1989.

/11/ M. Steinbrunn: "Optimierung der Zeitauflösung von NETMON-II im Hinblick auf den Einsatz in Transputer-Netzwerken", Studienarbeit am Institut für Telematik, Universität Karlsruhe, Februar 1990.

/12/ M. Roth: "Leistungsbewertung einer OSI-Transportprotokoll-Implementierung auf Transputer-Netzwerken", Diplomarbeit am Institut für Telematik, Universität Karlsruhe, August 1990.

/13/ M. Zitterbart: "Parallel Protocol Implementations for High Speed Networks", SBT/IEEE International Telecommunications Symposium, September 1990, Rio de Janeiro, Brasilien.