DK 681.32: 621.395.742: 621.395.01

Betriebsabläufe in Anlagen mit Rechnern

Bernhard Walke

Einleitung und Zusammenfassung

Moderne Industrie- und Nachrichtentechnik ist heute ohne elektronische Rechenanlagen nicht mehr denkbar. Sie übernehmen dort die bisher vom Menschen wahrgenommenen Dienste wie Organisation und Abrechnung, Koordination, Steuerung und Regelung, Optimierung, Überwachung etc. und übertreffen dabei den Menschen an Ausdauer, Genauigkeit, Zuverlässigkeit und Wirtschaftlichkeit. Außerdem ersetzen Rechner viele bis vor kurzem noch in analoger Technik realisierte nachrichtentechnische Geräte. Neben den durch die Fortschritte der Großintegration und die daraus resultierenden Preis- und Leistungsverhältnisse gegebenen sehr günstigen Voraussetzungen verdanken die Rechenanlagen diesen Durchbruch auch der modernen Systemtechnik. Sie umfaßt Arbeitsgebiete wie

- Software-Technologie (Entwurfs-, Realisations-, Test-, Meß- und Wartungsmethoden für Software),
- Systemplanung (Aufgabenaufteilung, Dimensionierung und Zusammenspiel von Systemteilen, Zuverlässigkeits- und Sicherheitstechnik und anderes),
- Verkehrstheorie (Leistungsbestimmung durch Berechnung und/oder Simulation von Modellen, Optimierung von Systemen oder Systemteilen, Engpaßanalysen usw.).

Der vorliegende Beitrag berichtet über systemtechnische Arbeiten, bei denen moderne Methoden und Werkzeuge eingesetzt wurden. Dazu werden beispielhaft einige Arbeitsthemen

mit Problemstellung, Lösungsverfahren, angewandten Werkzeugen und Ergebnissen beschrieben.
Aus dem Gebiet der *Verkehrstheorie* werden das Problem der *Echtzeitbearbeitung* unter Beachtung von *Terminen* behandelt und Ergebnisse eines Beispiels vorgestellt. Außerdem werden einige der neuentwickelten Faustregeln für Anwendungsprogrammierer genannt. Sie haben für die Ablaufsteuerung und Programmierung von Prozeßrechnern erhebliche Bedeutung als Voraussetzung für den effektiven Rechnereinsatz.
Ein weiteres Beispiel beschreibt den Aufbau und die Funktion eines für die AEG-80-Prozeßrechner entwickelten *Software-Meßsystems*. Dieses Werkzeug eignet sich als Hilfsmittel für alle drei oben genannten Arbeitsgebiete der Systemtechnik. Es folgt als systemplanerische Aufgabe die Knoten- und Leitungsdimensionierung eines *Paketvermittlungsnetzes*.
Ein softwaretechnologisches Beispiel schließt diese Übersicht ab und beschreibt einen *Umweltsimulator* für einen isolierten Paketvermittlungsknoten. Das Problem, diesem Knoten ein arbeitendes Netz zu simulieren, wird dadurch gelöst, daß ein weiterer Knoten simultan als Verkehrs-Generator und Verkehrs-Senke dient.
Die Untersuchungen betreffen in allen Fällen Probleme der Vorhaben von Fachbereichen verschiedener Unternehmensbereiche von AEG-TELEFUNKEN, für die das Forschungsinstitut Ulm gearbeitet hat.

1. Abläufe innerhalb von Rechnern

1.1. Berechnung einer parameterabhängigen Abfertigungsstrategie für den Echtzeitbetrieb von Prozeßrechnern

In vielen Anwendungsfällen von Prozeßrechnern ist es notwendig, daß an den Rechner gestellte Aufgaben innerhalb kurzer Zeitspannen bearbeitet werden. Man spricht dann von *Echtzeitbetrieb* des Rechners. Die Aufgaben können von anderen Rechnern gestellt sein oder durch bestimmte Ereignisse innerhalb bzw. außerhalb des Rechners angestoßen werden. Typisch für solche Anwendungsfälle ist, daß die Entstehungszeitpunkte der Aufgaben oft nicht im voraus bekannt sind. Aus der Sicht des Rechners treffen Aufgaben quasi zufällig bei ihm ein. Üblicherweise unterscheiden sich die Aufgaben sowohl in der beanspruchten Verarbeitungsleistung des Rechners als auch in der Dringlichkeit der Fertigstellung. Daraus resultiert die verkehrstheoretische Aufgabe, einen Algorithmus (Abfertigungsstrategie) zu bestimmen, der für jede denkbare Belastungssituation die Bearbeitungsreihenfolge wartender Aufgaben so festlegt, daß ein bestmögliches, den jeweiligen speziellen Umgebungsbedingungen angepaßtes Verhalten des Rechners folgt.

Wir betrachten den Fall, daß eine vom Rechner in Arbeit genommene Aufgabe nicht unterbrochen wird. Die Wartezeit w der Aufgabe zwischen den Zeitpunkten der Ankunft und des Bearbeitungsbeginns ist ein Maß für das Echtzeitverhalten des Rechners. Ist die durchschnittliche Belastung des Rechners mit Aufgaben (durch den jeweiligen Anwendungsfall) vorgegeben, dann ist die Minimierung der Wartezeiten ein denkbares Optimierungsziel. Liegen unterschiedlich wichtige Aufgaben vor, so können diese durch unterschiedliche Forderungen für die Wartezeit gekennzeichnet werden. Das Optimierungsziel ist dann die möglichst gute Einhaltung (bzw. Unterschreitung) der einzelnen geforderten Wartezeiten. Dieses Problem wurde an Hand eines Rechnermodells (Bild 1) bearbeitet. In diesem Modell sind Aufgaben der gleichen Wichtigkeit zusammengefaßt und durch Indizierung gekennzeichnet. Es gibt N verschiedene Wichtigkeiten – wir sprechen von N Typen von Aufgaben. Jeder Typ hat eine individuelle Ankunftsrate λ_i $(1 \leq i \leq N)$ und eine Bediendauer, die durch eine beliebig wählbare Verteilungsfunktion festgelegt ist. Die Warteschlangen muß man sich als im Zentralspeicher eines

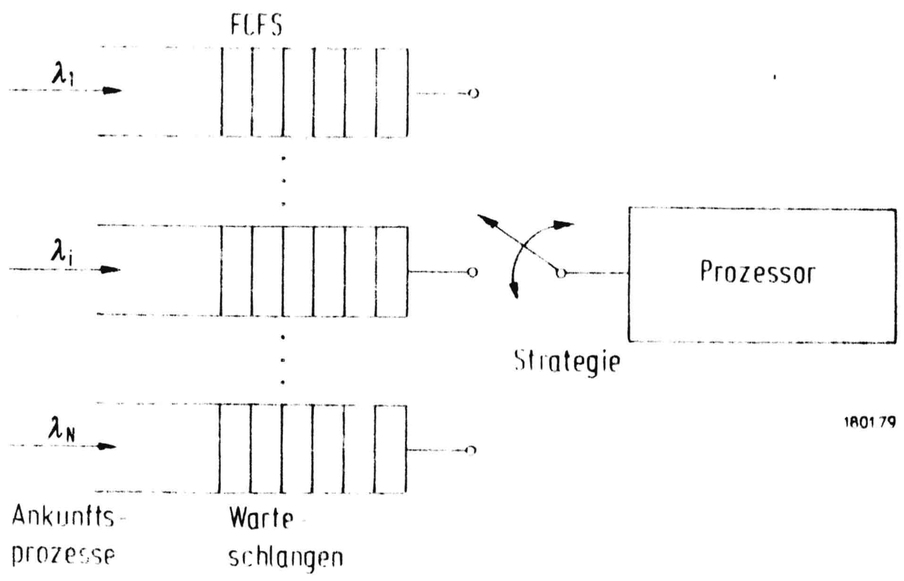

Bild 1. Verkehrsmodell mit N Typen von Anforderungen

Rechners existierende Verwaltungseinheiten vorstellen. Der Rechnerkern (Prozessor) benutzt eine gegebene Abfertigungsstrategie, um nach Fertigbearbeitung einer Aufgabe unter den wartenden die nächste zur Bearbeitung auszuwählen. Innerhalb jeder Warteschlange befinden sich nur gleich wichtige Anforderungen; deshalb wird dort in der Reihenfolge der Ankunft (**f**irst **c**ome **f**irst **s**erve, FCFS) abgearbeitet.
Die berechnete Strategie ermittelt im Bedarfsfall die aktuellen *Prioritäten* $q_i(t)$ aller Aufgaben des Typs i $(1 \leq i \leq N)$, die am prozessor-nächsten Platz der Warteschlangen stehen, vergleicht sie und wählt die Aufgabe mit kleinster Prioritätszahl $q_i(t)$ aus. Dabei folgen alle Prioritäten dem Gesetz $q_i(t) = \omega_i - w_i(t)$, wobei ω_i die einzuhaltende (bzw. zu unterbietende) Wartezeit von Aufgaben des Typs i und $w_i(t)$ die aktuelle bisher für eine Aufgabe verstrichene Wartezeit ist. ω_i ist ein Maß für die Dringlichkeit der Bearbeitung einer Aufgabe des Typs i und wird *Wichtigkeit* genannt. Der Verwaltungsaufwand dieser Strategie kann klein gehalten werden [1], [8]. Die Wichtigkeiten ω_i sind Parameter, über die sich die Wirkung der Strategie stark beeinflussen läßt: sind alle Wichtigkeiten gleich, dann resultiert für alle Aufgaben gemeinsam die *Strategie* FCFS. Sind die Wichtigkeiten sehr verschieden, dann wird als Grenzfall eine bei Prozeßrechnern schon übliche Strategie angenähert: Bei ihr ist die Priorität $q_i(t) = \omega_i$ kon-

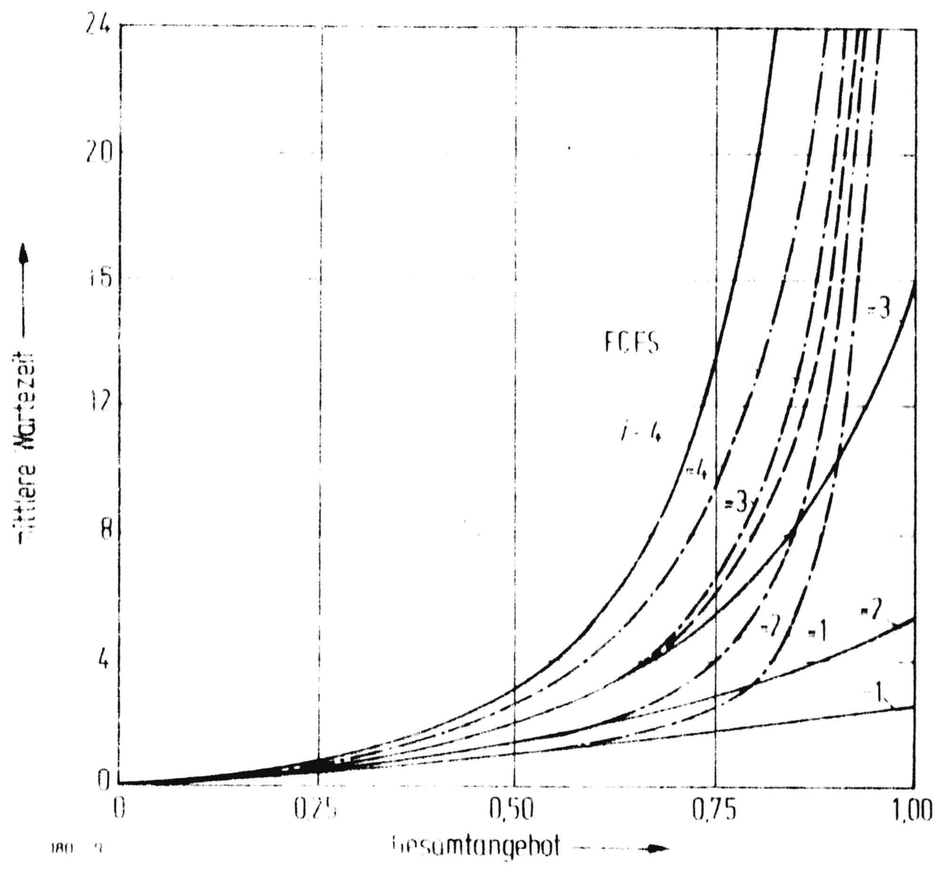

Bild 2. Terminorientierte Abfertigung von vier Typen von Anforderungen
alle Typen gleich wichtig
sehr unterschiedliche Wichtigkeit
etwas verschiedene Wichtigkeit

stant, unabhängig von der aktuellen Wartezeit; man spricht hier von *statischen* Prioritäten.

Bild 2 zeigt die berechnete mittlere Wartezeit eines Beispiels mit $N = 4$ Typen von Aufgaben in Abhängigkeit von der Belastung des Rechners *(Gesamtangebot)*. Dargestellt sind die Fälle

- gleiche Wichtigkeit aller Aufgaben (gestrichelt, FCFS),
- *sehr* verschiedene Wichtigkeiten der vier Typen von Aufgaben (ausgezogen),
- *etwas* verschiedene Wichtigkeiten (strichpunktiert).

Die einzelnen Kurven zeigen das Ergebnis für jeweils einen Typ *i* von Aufgaben.

Der Anwender interessiert sich nicht nur für die mittlere Wartezeit, sondern auch für ihre Verteilung. Denn daraus kann er ablesen, mit welcher Wahrscheinlichkeit bestimmte Wartezeiten auftreten. Ein neu entwickeltes Näherungsverfahren [2] erlaubt die Berechnung dieser Verteilungen für jeden Typ von Aufgaben und liefert damit alle notwendigen Unterlagen, wie sie der Systemplaner benötigt.

1.2. Faustregeln für Programmierer

In vielen Anwendungsfällen von Prozeßrechnern ist die Summe der Befehle und Daten aller vorrätigen Programme zur Aufgabenbearbeitung sehr viel größer als der verfügbare Zentralspeicher des Rechners. Dann verwendet man Hintergrundspeicher (z.B. Wechselplatten) als zusätzliche Datenträger. Es ist dann zu entscheiden, ob (beim Aufruf mehrerer Programme gleichzeitig) die Befehle und Daten weniger Programme je *vollständig* in den Zentralspeicher geladen werden sollen, oder ob nur sinnvolle *Ausschnitte* des Befehls- und Datenvorrats eines jeden Programmes geladen werden. Die letztgenannte Lösung hat Vorteile insofern, als bei gegebener Zentralspeichergröße mehr Aufgaben quasi gleichzeitig bearbeitet werden können, ohne daß dabei die mittlere Wartezeit gegenüber vollständig geladenen Programmen zunehmen muß. Das liegt an einer plausiblen Eigenart von Programmen: Besteht die Bearbeitung einer Aufgabe aus einer Reihe von Teilaufgaben, dann spiegelt sich das im Befehls- und Datenvorrat eines Programms so wieder, daß *ein* Ausschnitt für *eine*, ein *anderer* für eine *andere* Teilaufgabe zuständig ist. Zur Erledigung einer Teilaufgabe benötigt man nur den zugehörigen Befehls- und Daten*ausschnitt*. Beim Fortschreiten der Bearbeitung müssen die nicht mehr nötigen Befehls- und Datenteile gegen andere, aktuell benötigte ausgetauscht werden. Der Bearbeitungsablauf eines Programms besteht dann aus einer Folge von Transport- und Rechenphasen und kann wie in Bild 3 modelliert werden. Mit der Rate λ (Aufgaben je Zeiteinheit) treffen Bearbeitungswünsche beim Rechner ein, werden nach ihrer Wichtigkeit unterschieden und verursachen zunächst das Laden von einigen für den Bearbeitungsbeginn wichtigen Befehlen und Daten vom Hintergrundspeicher in den Zentralspeicher. Der Rechnerkern arbeitet (gesteuert durch eine Strategie) an jeder Aufgabe solange, bis Befehle oder Daten aus dem Hintergrundspeicher ergänzt bzw. ersetzt werden müssen, so daß sich Transport- und Rechenphasen solange abwechseln, bis die gesamte Aufgabe bearbeitet ist.

Modellbetrachtungen und Messungen in Rechern haben ergeben, daß Betriebssystem- wie auch Anwendungs-Programmierer die Möglichkeiten dieser Technik sehr oft falsch einsetzen. Das typische Ergebnis davon ist, daß auf eine sehr kurze Rechenphase einer Aufgabe eine im Mittel sehr lange Transportphase folgt, so daß unter dem Strich zwar sehr viele Aufgaben gleichzeitig im Zentralspeicher vertreten sind, der Rechnerkern aber sehr oft leersteht, weil alle Aufgaben gerade transportieren wollen, um ihre Befehls- und Datenbestände im Zentralspeicher zu aktualisieren. Bei unveränderter Belastung eines Prozeßrechners wird für jeden Typ von Aufgaben die mittlere Wartezeit umso größer, je größer das Verhältnis

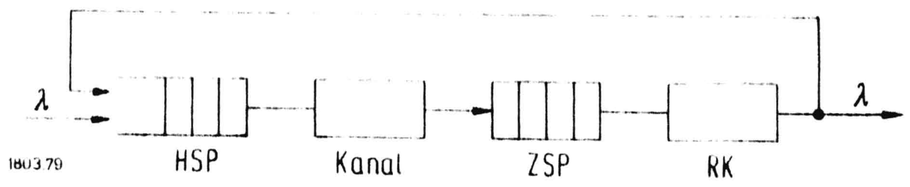

Bild 3. Warteschlangen-Netz-Modell eines Realzeitrechners mit Ankunfts- und Fertigstellungsrate λ (Anforderungen pro Zeiteinheit), unbegrenzt großem Hintergrundspeicher HSP, einem Transportkanal, Zentralspeicher ZSP mit *n* Plätzen und einem Rechnerkern RK. Folgt auf eine RK-Phase eine Transportphase, dann wird der Rückkopplungsweg zum HSP benutzt

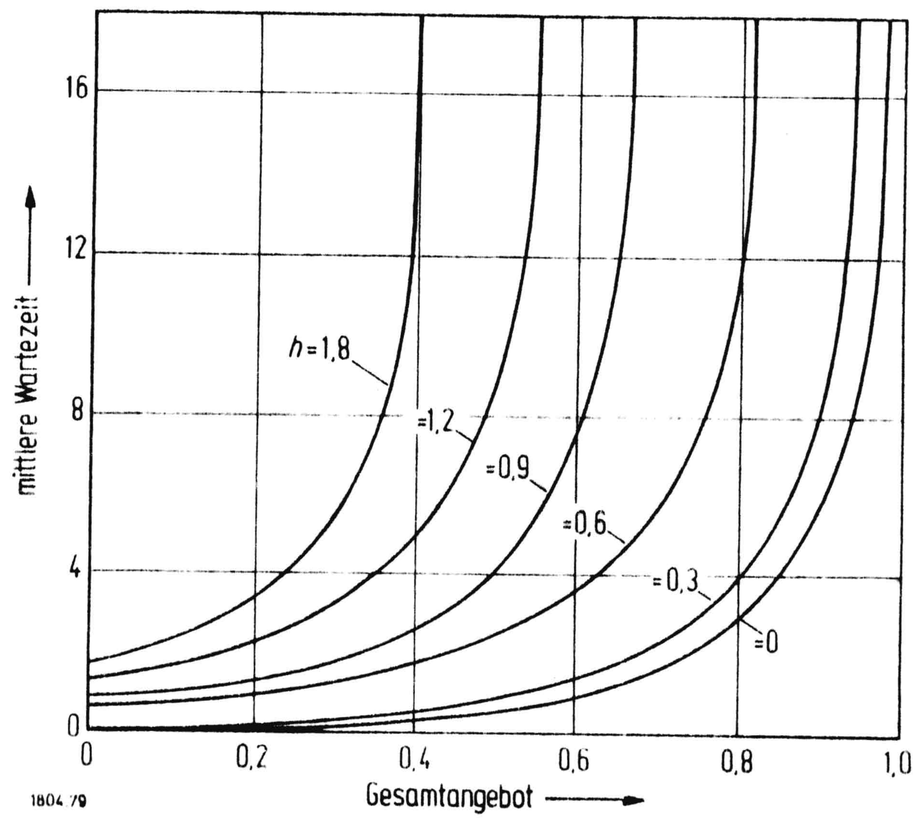

Bild 4. Mittlere Wartezeit über der Belastung des Rechnerkerns. Parameter ist das Transportzeit/Rechenzeit-Verhältnis *h*

h = (mittlere Transportphasendauer/mittlere Rechenphasendauer) ist (vergl. Bild 4). Je mehr Aufgaben gleichzeitig im Zentralspeicher mit Befehls- und Datenausschnitten vertreten sind, um so größer wird h.

Als Faustregeln können folgende Forderungen abgeleitet werden [7]

- h sollte deutlich kleiner als 1 sein (z.B. $h = 0{,}7$)
- es nutzt wenig, viele Programmausschnitte gleichzeitig im Zentralspeicher zu halten, selbst wenn h klein ist; deshalb sollten etwa drei Programme gleichzeitig als Richtwert angesehen werden
- bei der Programmierung sollte darauf geachtet werden, daß zusammengehörige Befehle im gleichen *Segment* (= Verwaltungseinheit für größere Datenmengen) liegen; das gleiche gilt für Daten
- bis auf Ausnahmefälle muß man davon ausgehen, daß ein Programmausschnitt etwa die Hälfte (oder mehr) aller Befehle und Daten eines Programmes beinhalten muß, sonst läßt sich $h < 0{,}7$ nicht erreichen,
- hat man keine sehr detaillierten Vorkenntnisse über das Verhalten der Programme, insbesondere die unterschiedlich häufige Nutzung der Befehle und Daten und den Platzbedarf für einen Ausschnitt, der die Forderung $h < 0{,}7$ erfüllen kann, dann sollte man auf das Laden von Programmausschnitten in den Zentralspeicher verzichten und statt dessen (soweit das geht) vollständige Programme laden.
- Transporte wegen nur teilweise geladenen Programmen beanspruchen Transportzeit, die voll als Wartezeit wirksam ist.

1.3. Entwicklung eines Software-Meßsystems

In den vorausgegangenen Abschnitten war mehrfach die Rede davon, daß Vorkenntnisse über

- Details der Aufgabenbearbeitung
- das Programmverhalten

- die Dauer von Betriebsmittelbelegungen (Kanäle, Rechnerkern, Zentralspeicher),

und anderes mehr nützlich oder erforderlich sind.
Außerdem benötigt man (ähnlich wie Oszillographen, Meßsender etc. bei der Hardware-Entwicklung) Meßvorrichtungen zur Überwachung von Software-Abläufen. Man fordert Meßeinrichtungen nicht nur für die Software-Entwicklung, sondern auch für die spätere Anwendung eines Systems, um ständig Leistungsdaten, beispielsweise über Auslastung, Zeitverhalten und Belegungsdauern verfügbar zu haben. Solche Aufgaben werden von einem für die AEG-80-Prozeßrechner entwickelten, universellen, zentralen Software-Meßsystem wahrgenommen [3]. Bild 5 zeigt die prinzipielle Funktion: Ein Meßprozeß stellt den Kontakt zwischen Benutzer und Meßsystem her und erlaubt die Steuerung eines Meßprogramms (etwa Normieren, Ausgabe veranlassen, Meßpunkt einrichten/löschen). Das Meßprogramm ist ein Unterprogramm, das durch Meßpunkte (bei auf sie zeigendem Befehlszählerstand) angesprungen wird und aus der Parameterversorgung sowohl den jeweiligen Meßpunkt als auch die geforderte Messung entnehmen kann.

Daß Meßprogramm führt also meßpunktspezifisch Auswertungen durch. Ein Meßpunkt schließlich kann an fast beliebiger Stelle in der Software eingefügt werden, besteht aus einem Unterprogrammsprung in das Meßsystem und einer Parameterversorgung und eignet sich dazu, die Registration von Ereignissen anzustoßen, die an der für den Meßpunkt ausgewählten Stelle in der Software (Befehlszählerstand) verfügbar sind oder durch ein individuell vorzusehendes Hilfsprogramm bestimmbar sind.

Bild 6 zeigt den schematischen Ablauf im Meßprogramm. Durch eine Sperre kann das Meßprogramm kurzgeschlossen werden und nimmt dann keine Registrierungen vor. Andernfalls wird der vom Meßpunkt bezeichnete Meßwert nach einer der Meßmethoden (Zählen, Mittelwert, Trace, Verteilungsfunktion) ausgewertet, bevor der Rücksprung zum aufrufenden Meßpunkt erfolgt. Meßpunkte können einzeln aktiviert und deaktiviert werden. Zwischen- und Endergebnisse können lesbar aufbereitet ausgegeben werden. Der Zeitaufwand für einen Meßprogramm-Aufruf beträgt bei den Rechnern AEG-80–40/60 etwa 150 µs, bei der leistungsschwächeren AEG-80-20 etwa 250 µs. Der Gesamtumfang von Befehls- und Datenbereich beträgt etwa 9 K-Byte, wovon etwa 3 K-Byte resident im Zentralspeicher liegen müssen. Bisher wurde das AEG-80–40/60-Meßsystem zu Untersuchungen der Leistung der Zentralspeicher-Verwaltung (Kachelverwaltung) und für Messungen des Zeitbedarfs von Systembefehlen (SVC), also bei der Grundsoftware-Entwicklung eingesetzt. Es ist vorgesehen, ausgewählte Anwendungsfälle von Prozeßrechnern in Fallstudien zu

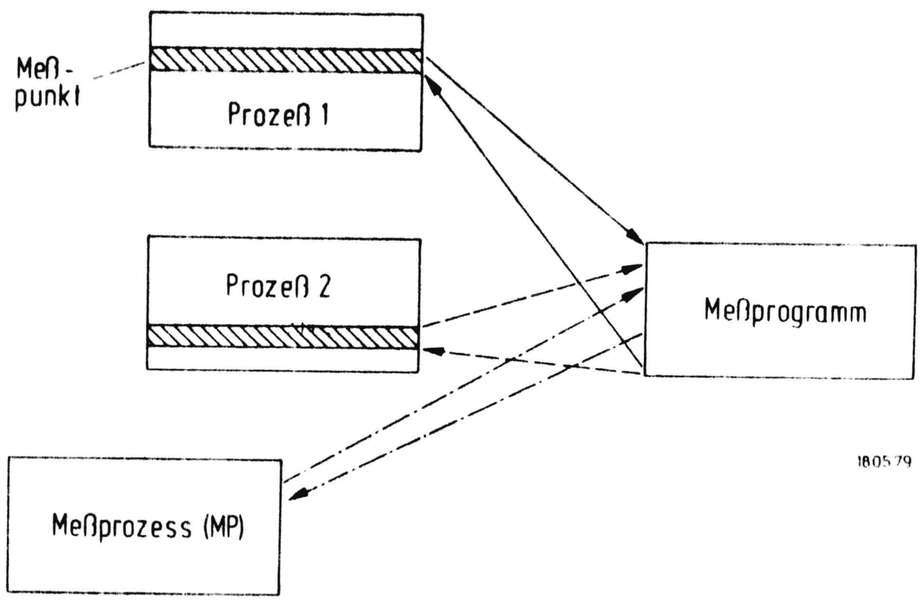

Bild 5. Struktur des Meßsystems. Der Aufruf des Meßprogramms erfolgt von Meßpunkten, die zwei oder mehr Befehle umfassen

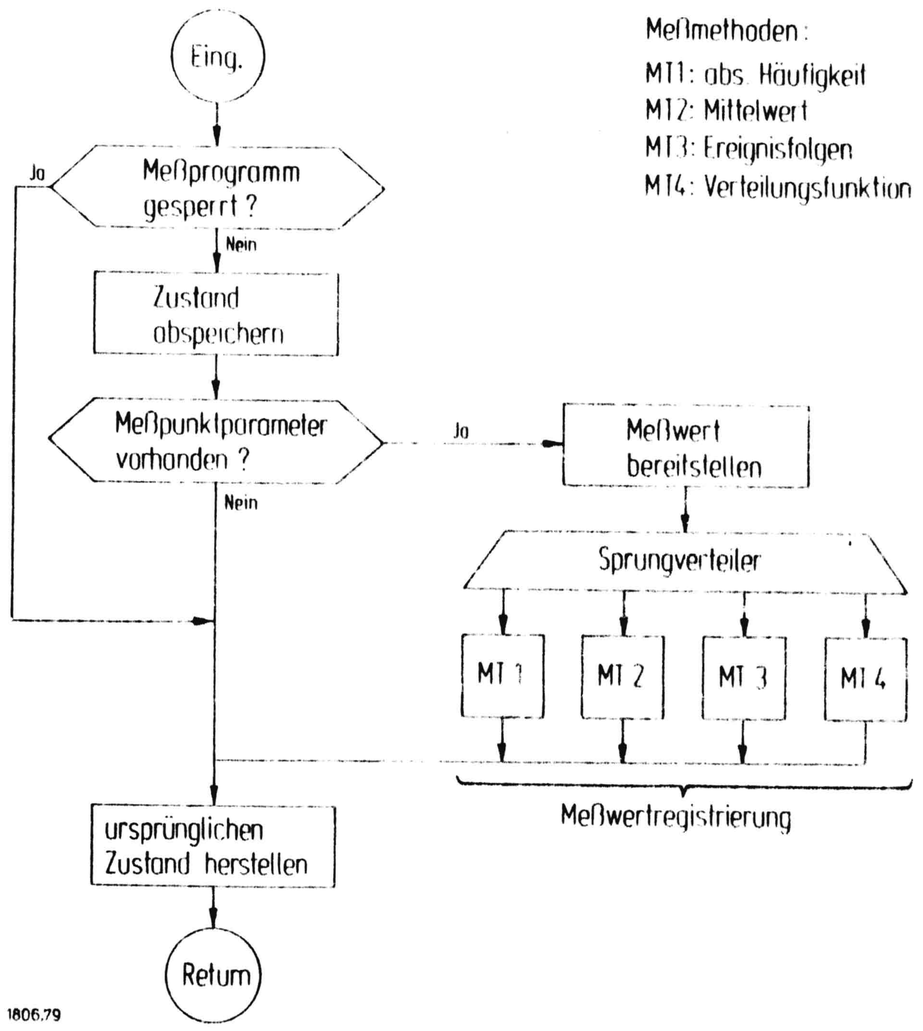

Bild 6. Schematischer Ablauf im Meßprogramm

analysieren, um einen Einblick in Ablauf-Details zu erhalten. Solche Untersuchungsergebnisse sind als Hilfsmittel zur Optimierung und zur leistungsgerechten Dimensionierung unerläßlich und werden schon lange gefordert [4].

2. Abläufe in Systemen mit Rechnern

Die bisher beschriebenen Methoden zur Optimierung, Leistungsbestimmung, Dimensionierung und Messung von einzelnen Rechnern lassen sich ebenso auch auf Systeme *von* oder Anlagen *mit* Rechnern anwenden. Es sind dazu beispielsweise Aussagen über Schwachstellen, Zeitverhalten, Leistung, Auslastung und andere Kenngrößen möglich. Als *erstes Beispiel* (Abschnitt 2.1) wird die Projektierung eines Paketvermittlungsnetzes beschrieben, bei dem strenge Zeitforderungen für die Nachrichtenübermittlung vorlagen. Ein *zweites Beispiel* (Abschnitt 2.2) beschreibt die Umgebungssimulation am Beispiel eines Vermittlungsknotens des genannten Netzes. Die Umgebungssimulation für ganze Systeme erfordert unter Umständen erheblichen Entwicklungsaufwand, wird aber zunehmend verlangt. Das vorgestellte Beispiel beschreibt eine besonders aufwandsarme Lösung dieses Problems.

2.1 Knoten- und Leitungsdimensionierung in einem Paketvermittlungsnetz

Bei der Planung eines Paketvermittlungsnetzes mit der genormten CCITT-Schnittstelle X.25 [6] treten ähnliche Aufgaben auf, wie sie auch bei der Dimensionierung eines Rechners zu lösen sind. Die Verkehrslast und die Leistungsforderungen sind bekannt. Es ist festzulegen, welche Organisation im Vermittlungsknoten und welche Hardware-Ausrüstung (Zahl der Leitungen zwischen Knoten, Puffergröße, Rechnerkernleistung, Hintergrundspeicher und anderes) gewählt werden müssen, um geforderte Zeitbedingungen bei vorgegebenem Durchsatz zu erreichen.

Bild 7 zeigt schematisch ein Netz mit vier Vermittlungsknoten, wobei an jedem Knoten viele (beispielsweise 10) abgesetzte Konzentratoren und mehrere Rechenzentren (RZ) angeschlossen sind. Jeder Konzentrator verbindet Fernschreibgeräte (FS) und Datengeräte (vom Lochstreifen- bis zum Sichtgerät)

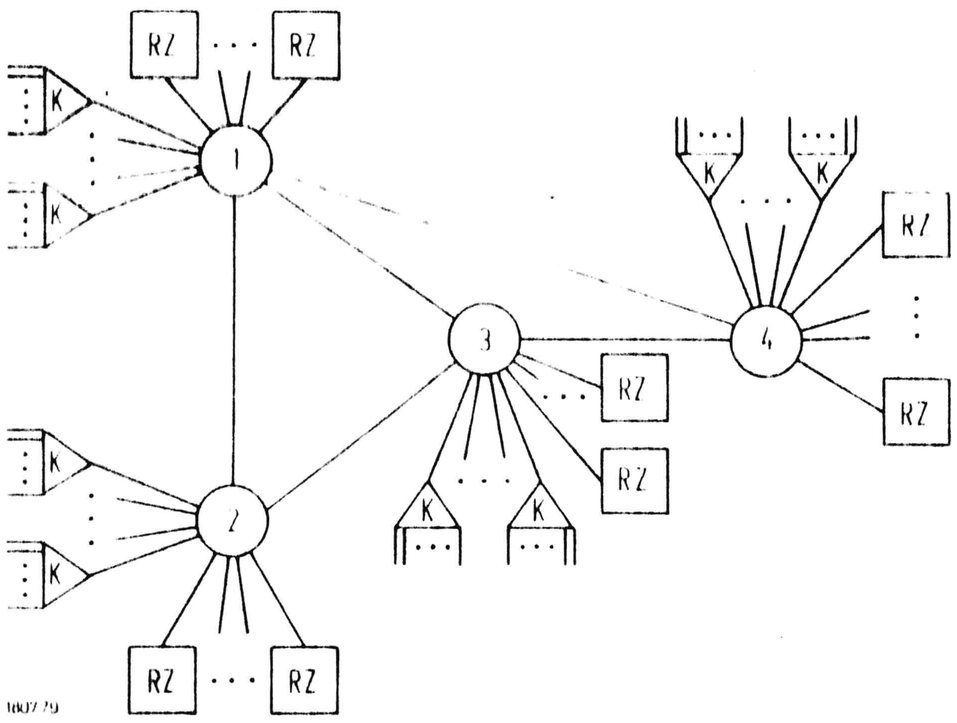

Bild 7 Paketvermittlungsnetz mit vier Vermittlungsknoten (1 bis 4). An jedem Knoten sind eine große Zahl von Konzentratoren (K) und Rechenzentren (RZ) angeschlossen

mit Leitungen einer Übertragungsleistung zwischen 50 und 2400 Bd mit dem Netz. Bis auf die Leitungen zwischen Konzentratoren und Geräten (außerhalb des Netzes) werden im Netz nur Vollduplex-Leitungen verwendet. Die Übertragungsleistung und benötigte Anzahl der Netzleitungen sollen festgelegt werden.

Der Fernsprechverkehr fließt zwischen lokalen FSn eines Knotens oder über das Netz. Beim Datenverkehr (D-Vk) gibt es Daten*dialog*-Vk, der zwischen Datenterminals und Rechenzentren abgewickelt wird und aus kurzen Fragen und Antworten besteht und Daten*stapel*-Vk. Der Stapelverkehr fließt von Datenendeinrichtungen (typisch: Lochstreifengeräten) in die Rechenzentren. Auch der Daten-Vk kann bezüglich eines Knotens lokal sein oder über das Netz gehen. Als Sendegeschwindigkeiten treten auf: bei FS typisch 50 und 100 Bd und bei Daten-Vk typisch 200 und 2400 Bd. Für wichtige Nachrichten sind maximale Übertragungszeiten zwischen Sende- und Empfangsgerät einzuhalten. Der Durchsatz eines Netzknotens beträgt einschließlich Verwaltungsdaten etwa 26 kBit/s und stammt zu 70% vom Stapel-Vk.

Für die Realisation eines Vermittlungsknotens kommen ein Prozeßrechner oder eine Anordnung aus mehreren kleinen Prozeßrechnern in Frage. Der genannte Datenstrom von 26 kBit/s ist, grob gesehen, für einen Prozeßrechner kein Problem, denn dort sind Datenraten von einigen Hundert kBit/s üblich. Im Unterschied zu bekannten Anwendungen von Prozeßrechnern mit hohen Datendurchsatzraten besteht der Datenstrom aus relativ kleinen Einheiten *(Paketen)* mit beispielsweise je 1 kBit, wobei für jedes Paket umfangreiche logische Arbeiten (Vermittlung, Quittung, Pufferung bis zum Quittungseingang, Zeitüberwachung etc.) durchzuführen sind. Dazu wird Verarbeitungsleistung im Vermittlungsknoten gefordert, die (nach sorgfältiger Aufschlüsselung des Aufwandes für die einzelnen Phasen der Bearbeitung) der Abarbeitung von im Mittel 11000 Befehlen je durchgesetztes Paket entspricht. Dabei sind Sonderdienste des betrachteten Netzes schon berücksichtigt. Neben der reinen Paketvermittlung fällt bei solchen Netzen weitere Arbeit an durch Auf- und Abbau virtueller Verbindungen, dynamische Flußkontrolle, Statistik, Netzwerkmanagement, Sonderdienste und anderes. Die geforderten Leistungen sollten preisgünstig sein, deshalb mußte sehr sorgfältig dimensioniert werden.

2.1.1. Im *ersten Schritt* werden *Mittelwert-Betrachtungen* angestellt, um ein Grobkonzept zu erhalten. Eine Abschätzung über den ständig im Vermittlungsknoten nötigen Teil der gesamten Software (Betriebssystem und Vermittlungsprogramme) ergibt, daß in diesem Anwendungsfall ein Prozeßrechner AEG 80-60 mit 32 Bit Wortbreite benötigt wird. Die dort verfügbare Verarbeitungsleistung mit etwa 2 µs je Befehl ermöglicht bei 11000 Befehlen je Paket und 40prozentiger Auslastung der Zentraleinheit allein durch Paketweitergabe einen Paketdurchsatz von 18 Paketen/s. Um die geforderten 26 kBit/s in Form von Paketen durchsetzen zu können, muß eine *Paketgröße* von mindestens 1,4 kBit gewählt werden.

Die Konzentratoren müssen unterschiedlichste Gerätekonfigurationen mit verschiedenartigsten Übertragungsmodi (synchron/asynchron) und Geschwindigkeiten zulassen. Deshalb wurde die Realisation durch je einen Prozeßrechner AEG 80-20 gewählt. Damit sind die Konzentratoren Bestandteil des Paketvermittlungsnetzes: Dort werden Nachrichten in Pakete zerlegt und aus Paketen wieder zusammengesetzt.

Neben der Durchsatz-Forderung von 26 kBit/s bestehen Zeitforderungen: Ein Dialogabschnitt mit der Länge 500 Zeichen/Frage und 500 Zeichen/Antwort soll in 90% aller Fälle in höchstens 10 s vom Gerät zum Ziel-RZ (Frage) und zurück (Antwort) übertragen werden. Dafür sind Zeitbetrachtungen nötig wie im folgenden Beispiel gezeigt wird.

Annahmen: 1000 Zeichen (Frage und Antwort) ergeben beim ISO 7-Bit-Code mit (7 + 1 Parity) Bits bei 1536 Bit/Paket 5,2 Pakete je Dialogabschnitt. Mit 6 Byte je Paketkopf ist die Übertragungsdauer auf einer 2,4-K-Bd-Leitung für ein Paket $T_U = 660$ ms. Es wird die Übertragung eines Dialogs vom Gerät zum Konzentrator, Knoten und RZ (dort über 2,4-kBd-Leitungen) betrachtet. Konzentrator und Knoten können ein Paket erst weitergeben, wenn es vollständig vorliegt. Das erste Paket jeder zusammenhängenden Folge von Paketen wird deshalb in Konzentrator, Knoten und RZ je für die Dauer $T_s = T_U$ aufgehalten, bevor es an den nächsten Empfänger weitergegeben wird (im RZ ist der Empfänger am gleichen Ort). Eine durchgeschaltete Verbindung würde nur *eine* Zeit T_U benötigen. Im Paketnetz treten für die Dialog-Frage *zwei* Zeiten T_s zusätzlich auf. Alle nachfolgenden Pakete folgen nahtlos, sind aber gegenüber dem Vorgänger an jedem denkbaren Ort im Netz um die Zeit $T_s = T_U$ verzögert (Tabelle 1).

Die so bestimmte Leitungs-Übertragungsdauer vergrößert sich noch durch Wartezeiten vor belegten Leitungen und Prozessoren um die Zeit T_W, so daß die Frage die Zeit $T_F = 2\,T_s + 1{,}2\,T_U + T_W$ benötigt. Die Antwort hat vier Pakete und benötigt ebenfalls zwei Aufsammelzeiten T_s (in Knoten und Konzentrator) sowie die Übertragungsdauer $4 \cdot T_U$. Einschließlich der Wartezeit T_W ergibt sich für die Antwort $T_A = 2\,T_s + 4\,T_U + T_W$. Demnach benötigt die Dialognachricht die Zeit $T_D = T_F + T_A = 4\,T_s + 5{,}2\,T_U + 2\,T_W = 6{,}09\,\text{s} + 2\,T_W$.

Tabelle 1. Zeitdiagramm für eine Frage mit 1,2 Paketen Umfang

Zeit t	0	T_s	$2T_s$	$2T_s + T_U$	$2T_s + 1{,}2T_U$
Kommentar*)	1	2	3	4	5

*) Kommentar 1: Bildschirm voll zur Ausgabe
2: 1. Paket im Konzentrator
3: 1. Paket im Knoten; 2. Paket noch im Konzentrator
4: 1. Paket im RZ; 2. Paket noch im Knoten
5: 2. Paket (nur 20% der Paket-Größe) im RZ

2.1.2. Im *zweiten Schritt* wird ein Modell eines Knotens entwickelt, um den Einfluß der statistisch schwankenden Leitungs- und Knotenbelastungen auf die Wartezeit T_W und damit die Dialogübertragungszeit T_D zu ermitteln.

Neben genaueren Aussagen über das Zeitverhalten (als es die Mittelwert-Betrachtung zuläßt) wird angestrebt, den Bedarf

an Puffer für Pakete im Knoten zu ermitteln. Solche Aussagen erhält man durch Berechnung oder Simulation eines Knoten-Modells. Bild 8 zeigt die Struktur, wobei nur drei Typen ankommender und abgehender Leitungen gezeichnet sind, mit den zugehörigen Ein- und Ausgabe-Raten. Das Modell berücksichtigt die Anzahl der Leitungen je Typ, beispielsweise 10 ein/abgehende Konzentratorleitungen mit der jeweiligen Rate λ_{EKZ} bzw. λ_{AKZ}. Es umfaßt die Eingabe ($T_{\ddot{U}}$) an den Knoten von Konzentratoren, Nachbarknoten und RZ'n, die Pufferung (WSP), prioritätstreue Vermittlung (T_r), Behandlung von gemeinsamen Paketen einer Nachricht (T_r), die Flußkontrolle (T_r) einschließlich der „Fenster" der logischen Transportebenen von X.25 (vergl. [6]) und die Ausgabe (T_{KZ}, T_{KN}, T_{RZ}) an die jeweiligen Empfänger (Konzentratoren, Nachbarknoten und RZ'n; Bild 8). Diese werden nicht direkt nachgebildet, sondern ihre Eigenschaften (zusammen mit denen der zugehörigen Leitungen) analytisch ermittelt. Dazu benutzt man Wartemodelle entsprechend Bild 1 mit Prioritäten zur Nachbildung der Leitungen zum RZ, zum Nachbarknoten und zum Konzentrator und bestimmt dafür Erwartungswerte und Streuungen der zugehörigen Wartezeiten. Der Prozessor in Bild 1 entspricht *einer* Leitung, das Wartesystem enthält alle auf Ausgabe wartenden Pakete. Bei *mehreren* parallelen Leitungen gibt es entsprechend viele parallele Prozessoren, aber nur *ein* Wartesystem. Die resultierenden Wartezeiten T_{WL} vor den Leitungen ergeben zusammen mit den Leitungszeiten T_U die Zeiten T_{RZ}, T_{KZ} und T_{KN}. Diese Ergebnisse werden (unter Berücksichtigung der Streuung von T_{WL}) als Zufallszahlen entsprechender Ausgabedauer-Verteilungen angesehen. Das bedeutet, daß der unterhalb des Knotens gezeichnete Teil des Modells in Bild 8 durch Ergebnisse analytischer Modelle beschrieben ist.

Das Quittieren erhaltener Pakete von den Empfängern und das dadurch ausgelöste Löschen von Paketen im Knotenpuffer, die anderenfalls nochmals gesendet werden müßten, wird ebenfalls *analytisch* behandelt. Dabei muß der Quittungsverkehr von den Empfängern zum Knoten auf den Rückleitungen, insbesondere seine Verzögerungen durch Warten, betrachtet werden.

Der verbleibende Teil des Modells nach Bild 8 kann wegen der unterschiedlichen durch T_r erfaßten Dienste des Knotens nur durch *Simulation* untersucht werden. Dafür stand der in [5] beschriebene GPSS-FORTRAN Simulator als Bausatz zur Verfügung.

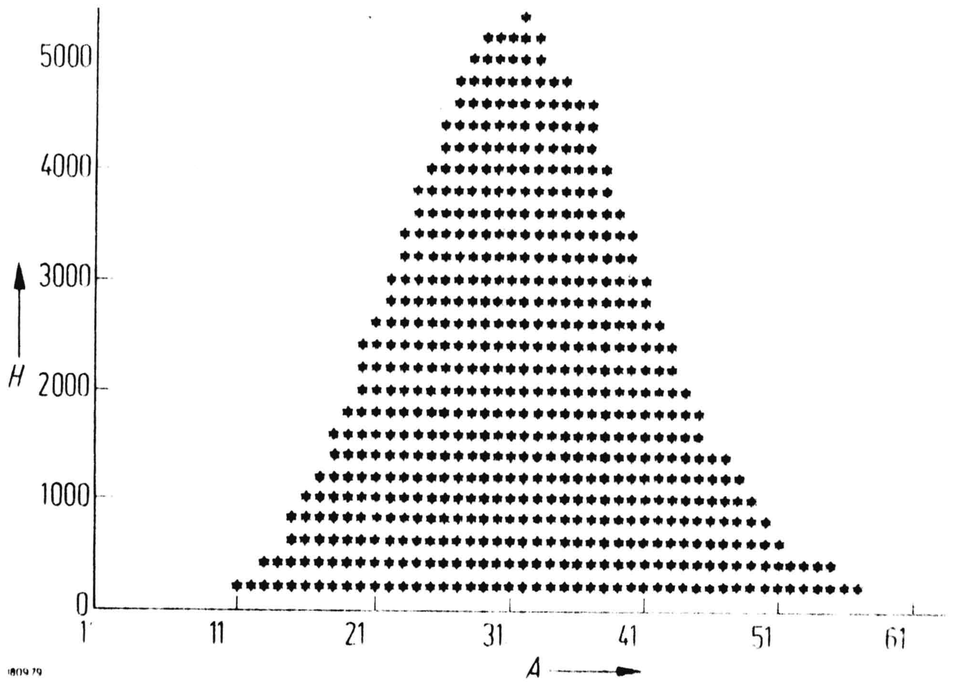

Bild 9. Platzbedarf-Verteilung für Pakete im Puffer des Vermittlungsknotens

Bild 9 zeigt den gesamten Pufferbedarf für Pakete im Vermittlungsknoten. Aus der Paketgröße ergibt sich der maximal nötige Platzbedarf. Bild 10 zeigt die Verweilzeit-Verteilungen für Dialog-Fragen aus schnellen und langsamen Quellgeräten bei 64 Zeichen/Prioritätspaket (Normalpakete haben hier die dreifache Größe). Dialog-Antworten sind gleich lang wie Dialog-Fragen unterstellt. Die Länge einer Dialog-Frage streut, deshalb hat sie bis zu 14 Pakete. Der Paket-Sende-Abstand für langsame Geräte beträgt im Modell 7s. Für solche Geräte ist die Verweilzeit-Verteilung für jede Länge einer Frage ablesbar, beispielsweise bei acht Paketen 52s. Ein entsprechendes schnelles Gerät hätte 0s Pausenzeit, benötigt also $(52 - 7 \cdot 7)$s $= 3$s als maximale Übertragungszeit. Frage und Antwort benötigen maximal die doppelte Zeit. Anstelle der für 90% aller Fälle geforderten 10s Übertragungsdauer für komplette Dialoge werden maximal 6s benötigt. Man beachte, daß gegenüber Abschnitt 2.1.1. die Paketgröße verkleinert wurde. Zahl und Übertragungsleistung der Leitungen gehen als Parameter in das Hybrid-Modell ein. Zeit- und Pufferbedarf resultieren (bei gegebener Bearbeitungsleistung im Knoten) daraus als Simulationsergebnisse.

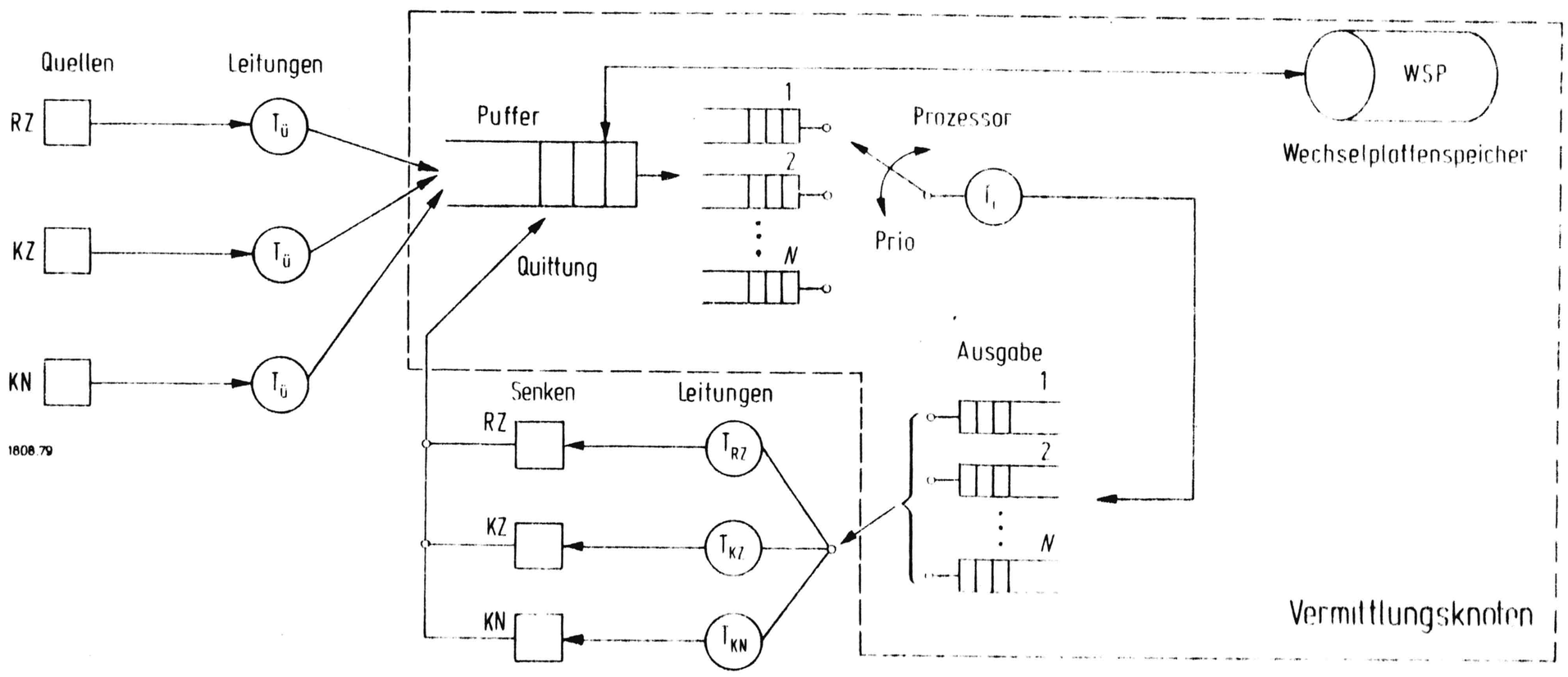

Bild 8. Struktur des Knoten-Modells. Es ist nur je ein Sender und Empfänger jedes Typs und eine zugehörige Leitung gezeigt. Auch ist die Differenzierung in Verkehrsarten nicht dargestellt, wird aber beachtet. Eingaben und Ausgaben (bis zum Eingang der Quittung) belegen Puffer

RZ = Rechenzentrum KZ = Konzentrator KN = Knoten T_U, T_r = Bearbeitungsdauern T_{RZ}, T_{KZ}, T_{KN} = Verweildauern

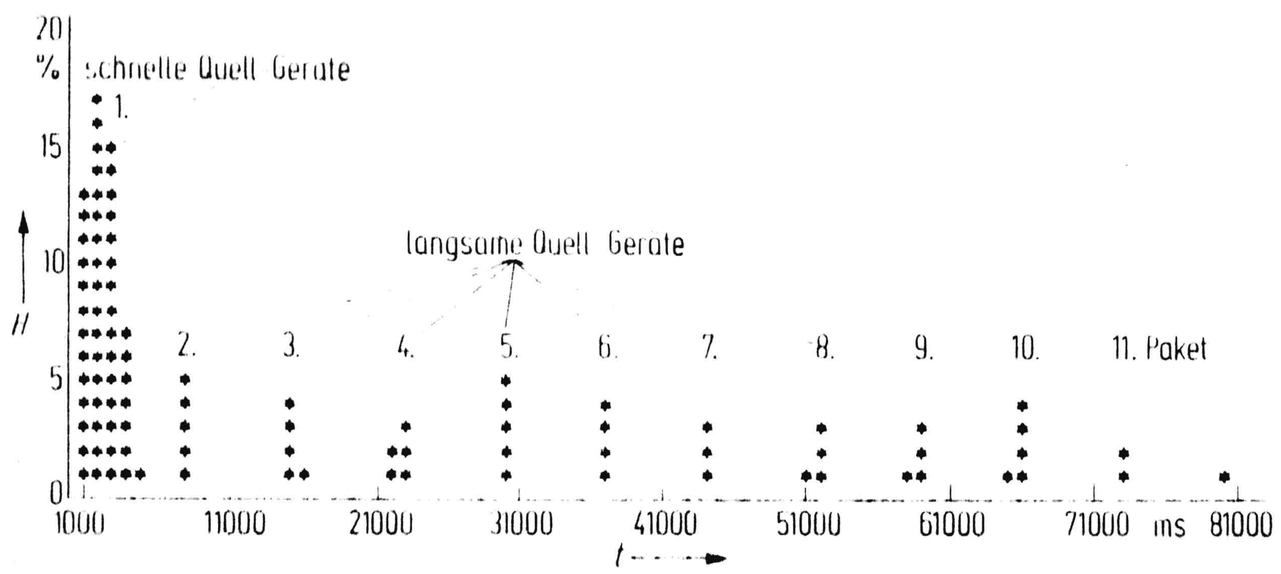

Bild 10. Übertragungsdauer-Verteilung für Fragen von Prioritäts-Dialogen in Abhängigkeit von der Anzahl von Paketen

2.2. Umgebungs-Simulation für einen Paketvermittlungsknoten

Die Umgebungs-Simulation für Rechner oder Rechner-Systeme schafft die Möglichkeit, ein fertiggestelltes System (oder Teile davon) mit (der wirklich vorgesehenen Belastung entsprechender) simulierter Last auf seine Leistungsfähigkeit zu prüfen. Solche Belastungen sind als Tests bei der Systementwicklung nützlich, in vielen Fällen neuerdings auch vom Kunden gefordert und deshalb notwendig.

Der Grundgedanke ist, daß Last- und Leistungs-Kenndaten eines entwickelten Systems bereits aus dem Pflichtenheft bekannt sind, so daß die Aufgabe bleibt, eine möglichst einfache Anordnung zu schaffen, die unter Verwendung vorhandener Schnittstellen reproduzierbar und leicht beeinflußbar den „Ernstfall" simuliert.

Als Beispiel wird die Umgebungssimulation eines Vermittlungsknotens des oben beschriebenen Netzes angeführt. Dabei wird jener Verkehr nachgebildet, der von Konzentratoren des Knotens, seinen Rechenzentren und von Nachbarknoten stammt. Anstelle des Anschlusses von Verkehrsquellen wie Konzentratoren, Rechenzentren und Nachbarknoten über entsprechende Leitungen werden diese Leitungen zu einem „Prüfrechner" AEG 80-60 geführt, der stellvertretend als Verkehrsquelle fungiert (Bild 11). Dieser Prüfrechner nimmt auch die Quittungen des Vermittlungsrechners sowie den von ihm weitergeleiteten Verkehr entgegen. Vier Programm-Moduln sind in diesem Zusammenhang speziell vorgesehen:

- Traffic-Generator
- Anpaßprozedur
- Traffic-Ziel
- Meßsystem.

Das unter Abschnitt 1.3 beschriebene Meßsystem wird zur Überwachung dynamischer Abläufe im Prüf- und Vermittlungsrechner eingesetzt.

Traffic-Generator

Zunächst besteht dieser Generator aus einem Modell, in dem alle betrachteten Verkehrsquellen getrennt verwaltet und (gemäß gegebener statistischer Beschreibungen von Nachrichtenlängen und des Zeitabstandes zwischen Nachrichtenübermittlungswünschen) Übertragungswünsche generiert werden. Lange Nachrichten werden (wie üblich) in mehrere Pakete mit zugehörigen Köpfen zerlegt. Zur Quellbeschreibung gehört eine Matrix mit Angabe der möglichen Empfänger und der Wahrscheinlichkeit jeder Verbindung dorthin. Jede Nachricht erhält ihre Priorität. Für die Verkehrserzeugung aus langsamen und schnellen Datenquellen sowie aus Fernschreibgeräten werden vorgegebene Verkehrswerte verwendet. Der Generator verfügt über einen Vorrat von Nachrichten-Köpfen (mit allen zu testenden Varianten). Von diesen wählt er je Nachricht einen aus, ermittelt nach gegebenen Zufallsgesetzen Empfänger und Länge der Nachricht, setzt sie im Paket-Kopf ein und füllt den Datenteil mit definierter, aber sonst wertloser Information. Das entsprechende Programm des Generators ist in FORTRAN geschrieben.

Die Nachrichtenübermittlungswünsche werden nicht in *Echtzeit*, sondern *vorab* erzeugt und stehen der Anpaßprozedur auf einem Wechselplattenspeicher zur Verfügung. Jede dort zur Sendung vorbereitete Nachricht ist gekennzeichnet durch Sendezeitpunkt (durch den Prüfrechner) und Nachrichtenkopf mit Eintragungen, die im Test zu prüfen sind. Entsprechend den realen Verhältnissen ist es möglich, Folgepakete mit einer Zeitverzögerung gegenüber dem ersten Paket der Nachricht (entsprechend der Sendegeschwindigkeit der angenommenen Quelle) zu bearbeiten. Die Reproduzierbarkeit der gesendeten Nachrichtenflüsse wird gewährleistet.

Der Verkehr wird so erzeugt, daß Ein- und Ausgabeleitungen des Vermittlungsrechners ähnlich belastet werden, wie im späteren Anwendungsfall. Daraus ergibt sich, daß die Aus- und Eingabeleitungen des Prüfrechners entsprechend belegt sind, so daß der Prüfrechner den für einen Vermittlungsknoten komplementären Verkehr hat.

Anpaßprozedur

Diese Prozedur beschafft sich die vom Traffic-Generator auf der Wechselplatte abgelegten Köpfe von Start- und Folge-Paketen einzelner Nachrichten mit den zugehörigen Sendezeitpunkten, und übergibt sie zum jeweiligen Sendezeitpunkt an den Prüfrechner. Dieser enthält die vollständige Software eines Vermittlungsknotens. Die Eingaben des Prüfrechners stammen von der Anpaßprozedur oder vom Vermittlungsrechner. Die Anpaßprozedur verwendet die gleichen Schnittstellen zur Vermittlungs-Software, wie sie vorhanden sind für Verkehr, der aus dem Netz von RZ'n oder Konzentratoren eingeht. Nach Übergabe der Pakete über diese Schnittstellen in die Vermittlungs-Software des Prüfrechners laufen normale Vermittlungsroutinen ab, die dazu führen, daß Ausgaben zu Konzentratoren des Prüfrechners, zu lokalen Rechenzentren und in das Netz entstehen. Die Ausgabeleitungen sind real vorhanden und führen (als Eingabeleitungen von den Konzentratoren, den lokalen Rechenzentren und dem Netz) vom Prüfrechner zum Vermittlungsrechner. Die Zahl dieser Leitungen ist ebenso groß, wie die wirklich am Vermittlungsrechner anzuschließende Anzahl.

Die Software des Vermittlungsrechners ist vollständig und wird für den Test nicht verändert. Lediglich die Richtungstabellen werden so gesetzt, daß der gesamte zu Nachbarknoten abgehende Verkehr zum Prüfrechner zurückfließt. Als Ausgabeleitungen zu den Konzentratoren des Vermittlungsrechners und seinen Rechenzentren fungieren entsprechende Eingabeleitun-

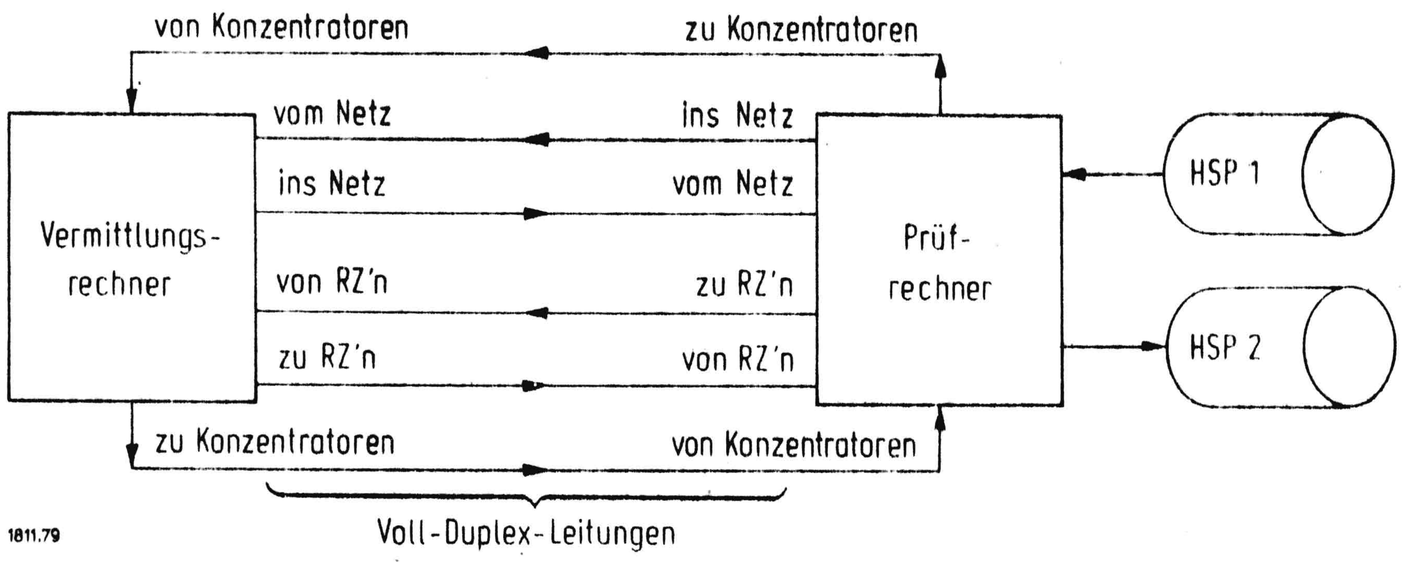

Bild 11. Umgebungs-Simulation eines Vermittlungsrechners
HSP 1: vom Traffic-Generator erzeugte Pakete mit Köpfen HSP 2: vom Vermittlungsrechner eingegangene Köpfe

gen des Prüfrechners. Bild 11 zeigt die zugehörige Anordnung. Für jede Verbindungsart (beispielsweise Knoten/Konzentrator) ist nur eine Verbindung gezeichnet. Die Hintergrundspeicher fungieren als Datenquelle (HSP 1) und Datensenke (HSP 2). Die Datensenke wird vom Modul „Traffic-Ziel" belegt.

Traffic-Ziel
Diese Prozedur verwendet die gleichen Schnittstellen wie die Anpaßprozedur, mit dem Unterschied, daß nicht Verkehr *eingespeist*, sondern im Prüfrechner *ankommender* Verkehr herausgezogen und für spätere Analysen auf einem Wechselplattenspeicher aufgehoben wird.
Jeder der beiden Rechner verfügt standardmäßig über das Meßsystem (vergl. Abschnitt 1.3). Zeit- und Auslastungsmessungen sind also wie im endgültig vorgesehenen Betrieb möglich. Da Prüf- und Vermittlungsrechner dieselbe Verkehrslast (komplementär) haben, der Prüfrechner aber zusätzlich durch die Anpaßprozedur und das Traffic-Ziel belastet wird, ist eine Entlastung des Prüfrechners nötig, um ihn nicht zum Engpaß werden zu lassen. Dabei läßt sich benutzen, daß alle Pakete leer sind, so daß rechnerinterne Transporte von Paket-Daten entfallen können. Weitere gegebenenfalls notwendige Vereinfachungen sind möglich durch Abschaltung aller nicht unmittelbar bei der Vermittlungsfunktion beteiligten oder benötigten Dienste.

NE3-UL
Eingang: 21.8.78

Die diesem Bericht zugrunde liegenden Arbeiten wurden mit Mitteln des Bundesministers für Forschung und Technologie gefördert. Der Bundesminister für Forschung und Technologie übernimmt keine Gewähr für die Richtigkeit, die Genauigkeit und Vollständigkeit der Angaben sowie für die Beachtung privater Rechte Dritter.

Schrifttum

[1] B. Walke: Prozeßrechner: statische und terminorientierte Prioritäten im Vergleich. Elektron. Rechenanl. (1978) Heft 6, S. 283–290
[2] B. Walke and W. Rosenbohm: Waiting-time distributions for deadline-oriented servicing (4th Int. Sympos. Comp. Syst. Modelling, Wien-Laxenburg, Jan. 1979) Proceed. by North Holland Publishing Company (1979)
[3] W. Rosenbohm: Ein Software-Meßsystem für Prozeßrechner. GMR-GI-GfK Fachtagung „Prozeßrechner 1977", Augsburg. Informatik Fachber. 7, Springer-Verlag, Berlin (1977), S. 513–524
[4] W. Heusler: Rationalisierung der Softwareerstellung. GMR-GI-GfK Fachtagung „Prozeßrechner 1977", Augsburg. Informatik Fachber. 7, Springer-Verlag, Berlin (1977), S. 39–61
[5] B. Schmidt: GPSS-FORTRAN, Einführung in die Simulation diskreter Systeme mit Hilfe eines FORTRAN-Programmpaketes. Informatik Fachber. 6, Springer-Verlag, Berlin (1977).
[6] W. Tietz: CCITT-Empfehlungen der V-Serie und der X-Serie: Datenübertragung. R. v. Deckers Verlag, G. Schenk, Heidelberg 1977
[7] B. Walke: Vergabe von Prioritäten und Einsatz von Betriebsmitteln in Realzeitrechnern: Übersicht und Entwurf einer Projektierungsrichtlinie. Wiss. Ber. AEG-TELEFUNKEN 52 (1979) 3
[8] B. Walke: Realzeitrechner Modelle, Theorie und Anwendung. R. Oldenbourg Verlag, Reihe Datenverarbeitung, München, Wien 1978