DK 681.31:65.015.13

Vergabe von Prioritäten und Einsatz von Betriebsmitteln in Realzeitrechnern – Übersicht und Entwurf einer Projektierungsrichtlinie

Bernhard Walke

Einleitung und Zusammenfassung

Hat man begrenzte oder vollständige Vorkenntnisse über Zeiten, Raten, Periodendauern usw., die im Zusammenhang mit der Programmbearbeitung bei Realzeitrechnern auftreten, dann kann man diese Kenntnisse nutzen, um bestimmte Betriebsziele zu erreichen. In der Regel sind solche Vorkenntnisse, die das Belastungsprofil eines Realzeitrechners beschreiben, bei Anwendungs-Programmierern vorhanden; manche der Daten sind auch schon im Pflichtenheft eines Anwendungsfalles beschrieben. Ohne Vorkenntnisse der deterministischen oder statistischen Eigenschaften des Belastungsprofils sind manche der nachstehend beschriebenen Methoden nicht anwendbar. Die hier beschriebenen Verfahren zur optimalen Ablaufsteuerung rechtfertigen den Aufwand, der möglicherweise bei der Ermittlung eines Belastungsprofils nötig ist; die Ergebnisse der unten besprochenen Beispiele sprechen für sich.

Ein Ziel der hier vorgelegten Richtlinie ist es, anzugeben, wie die Bearbeitungsreihenfolge konkurrierender Rechenprozesse sein muß, um vorgegebene Betriebsziele optimal zu erreichen. Dabei werden Aussagen über optimale Strategien der Prozessor-Zuteilung und die Vergabe von Prioritäten gemacht. Entsprechend den unterschiedlichen Anwendungsfällen der Praxis werden verschiedene Betriebsziele und die jeweils günstigste Prioritätenvergabe besprochen. Beispiele erläutern die zugehörigen Ergebnisse, wie man sie durch Berechnung analytischer Modelle erhalten kann. Anhand dieser Ergebnisse kann man durch Vergleich die Wirksamkeit verschiedener Prioritätsvergabe-Techniken kennenlernen.

Ein weiteres Ziel ist es, anhand von Beispielen zu zeigen, welche Nachteile man beim Abweichen von diesen optimalen Strategien zu erwarten hat. Außerdem wird gezeigt, wie sehr Hintergrundtransporte das Zeitverhalten eines Realzeitrechners beeinträchtigen. Deshalb sollte man zu einer möglichst transportarmen Programmierung übergehen.

Es wird zwischen „kritischem" und „unkritischem" Echtzeitbetrieb unterschieden. Im erstgenannten Fall dürfen Datentransporte zwischen Zentral- und Hintergrundspeichern *nicht* als Bestandteil der geforderten Verarbeitungsleistung auftreten. Die entsprechenden Aussagen gelten also für Systeme ohne Hintergrundspeicher und für zentralspeicher-residente Rechenprozesse in beliebigen Systemen. Im zweiten Fall, dem unkritischen Echtzeitbetrieb, treten während der Bearbeitung für den Fortgang notwendige Transporte zwischen Zentralspeicher und Hintergrundspeichern auf, wobei die konkurrierende Benutzung der Betriebsmittel *Zentralspeicher* und *Kanäle* durch verschiedene Rechenprozesse erheblichen Einfluß auf die Ergebnisse für die Betriebsziele haben kann. Hier sind also Strategien für den Einsatz auch dieser Betriebsmittel zu berücksichtigen. Anstelle von unkritischem Echtzeitbetrieb spricht man auch nur von Echtzeit- oder von Teilnehmer-Betrieb.

Wir unterscheiden Abfertigungsstrategien mit *statischen*, *dynamischen* und *internen* Prioritäten. Die Vorteile der Zusammenfassung von Anforderungen an unterschiedliche Programme in eine Gruppe mit gemeinsamen gleichen Betriebszielen werden dargestellt. Um diesen Text kurz zu halten, werden die benötigten Begriffe im Abschnitt 2 aufgelistet und nur kurz erläutert. Die angesprochenen Zusammenhänge zwischen *Belastungsprofil* eines Realzeitrechners und zugehörigem (vereinfachendem) Modell, Optimierungsziel und zugehöriger Abfertigungsstrategie und den jeweiligen durch analytische Rechnung oder Simulation gewonnenen *Ergebnissen* werden hier unter dem Gesichtspunkt der praktischen Anwendung beschrieben. Den mathematischen Hintergrund und insbesondere die Formeln zur Berechnung der dargestellten Ergebnisse sowie die detaillierte Beschreibung der genannten Strategien findet man in [1] bis [5] und den dort angegebenen Literaturstellen.

Die hier besprochenen Modelle, Strategien und Optimierungsziele und die zugehörigen Aussagen stützen sich auf sorgfältige Untersuchungen und eine umfangreiche Sichtung der Literatur über Realzeitrechner-Modelle [1]. Das Kapitel 5 beschreibt die wesentlichen Ergebnisse, die man in den letzten Jahren bei der Analyse von Realzeitrechnern mit Hintergrundspeichern gewonnen hat [4] und [6]. Berücksichtigt man die angegebenen Grundregeln, etwa für die

* Vergabe statischer Prioritäten,
* notwendigen Voraussetzungen für den effektiven Einsatz interner Prioritäten,
* Zuordnung von optimalen Strategien zu gegebenen Optimierungszielen,
* Programmierung bei Platzbeschränkung im Zentralspeicher in Systemen mit Hintergrundspeicher,

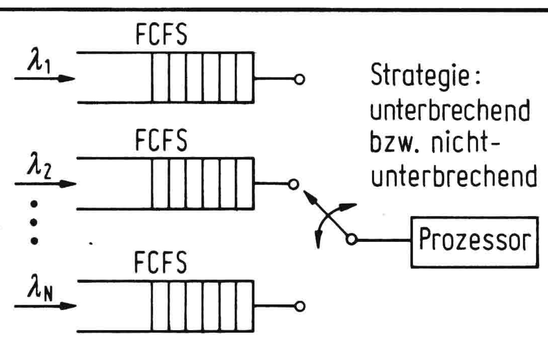
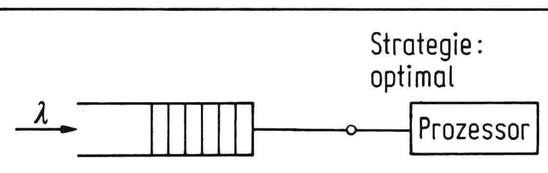
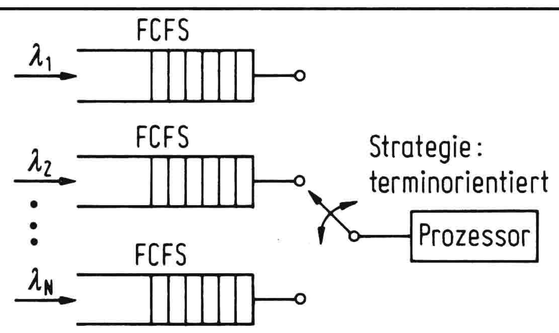
dann können grobe Organisationsfehler vermieden werden, und man kommt zu einer verkehrsgerechten Ablaufsteuerung. Das dabei resultierende System wird dann (im Rahmen der bei der Planung vorgesehenen Belastung) optimal. Da die aus dem Pflichtenheft entnehmbaren Daten über das Belastungsprofil eines Anwendungsfalles oft nur erste Schätzungen darstellen, sollte in jedem Fall nach der Realisierung eine mindestens grobe meßtechnische Untersuchung der Verkehrsflüsse zur Bestimmung ihrer Parameter durchgeführt werden. Solche Daten eignen sich einerseits noch für einige einfache nachträgliche Optimierungen (wie etwa Änderung der statischen Prioritäten), andererseits lassen sie sich unter Umständen auf ähnliche Projekte übertragen und vergrößern damit die Vorkenntnisse bei neuen Projekten.

1. Die wichtigsten Ergebnisse

Tabelle 1 gibt eine Übersicht über behandelte Modelle, Optimierungsziele und zugehörige optimale Strategien. Man kann aus den betrachteten Modellen mit nur einer Bedienstation unmittelbare Aussagen für Modelle mit mehreren Bedienstationen (z. B. Rechnerkern mit Hauptspeicher und Transportkanal mit Hintergrundspeicher) ableiten. Im folgenden werden Begriffe verwendet, die in Abschnitt 2 definiert sind.

(I) Verwendet man *statische* Prioritäten zur Ablaufsteuerung, dann sollte man die Prioritäten den durch *spontane* Ereignisse aktivierten Rechenprozessen so zuteilen, daß die *Priorität um so höher ist, je kleiner die mittlere Bediendauer* (= Rechenzeit) eines Rechenprozesses ist. Diese Regel gilt für *nicht-unterbrechende* Prioritäten uneingeschränkt. Sie gilt auch bei *unterbrechenden* Prioritäten, solange die Bediendauern jeder Priorität nicht stärker streuen als eine negativ exponentielle Verteilung. Bei stärkerer Streuung sollten interne (d. h. vom Betriebssystem gesetzte) Prioritäten verwendet werden.

Tabelle 1: Übersicht über Optimierungsziele, Modelle und Ergebnisse, die hier behandelt werden. Neben den eingezeichneten Modellen mit einer Bedienstation werden auch Ergebnisse für sequentiell arbeitende Bedienstationen (Rechnerkern und Transportkanal) besprochen.

Optimierungsziel	Modell	Ergebnisse	Bemerkungen
uneingeschränkte Bevorzugung von bestimmten Anforderungen gegenüber anderen. FCFS in jeder Prioritätsebene		optimale statische Prioritätenzuteilung bekannt, Gln. (1) u. (3). Berechnung der charakteristischen Größen für kritische Echtzeitanforderungen möglich	für kritische (ereignis- bzw. takt-gesteuert)/unkritische Echtzeit-anforderungen siehe Abschnitt 4.1.1 bzw. 4.1.2/5.3
wie oben, aber wartezeitoptimale Abfertigung in jeder Prioritätsebene		optimale Strategie (nur in Sonderfällen FCFS) bekannt [2] und mittlere Wartezeit berechenbar	ein Typ von Anforderungen je Prioritätsebene, Beispiel 4.2
wartezeitgünstige Abfertigung gleichberechtigter Anforderungen (mehrere Typen)		optimale Strategie hängt von Belastungsfall ab. Charakteristische Größen näherungsweise berechenbar oder Simulation	für kritische/unkritische Echtzeit-anforderungen siehe Abschnitt 4.2/5.2
bestmögliche Garantie von Terminen für *N* Typen von Anforderungen		optimale Strategie bekannt. Näherungsweise Berechnung der charakteristischen Größen für kritische Echtzeitanforderungen möglich	für kritische (ereignis- bzw. takt-gesteuert)/unkritische Echtzeit-anforderungen siehe Abschnitte 4.3.1 bzw. 4.3.2/5.4

2199.78

Diese Regel minimiert die gemeinsame mittlere Wartezeit aller Rechenprozesse und dabei etwa auch die mittlere Belegungsdauer von exklusiv reservierten Betriebsmitteln.
Abweichungen von dieser Regel können zu erheblichen Vergrößerungen der gemeinsamen mittleren Wartezeit führen und sind deshalb nur in begründbaren Fällen vertretbar. Für *periodisch* aktivierte Prozesse muß die Priorität um so höher sein, je kleiner die Periodendauer zwischen zwei Aufrufen desselben Prozesses ist. Die Prioritäten müssen unterbrechend sein. Man erreicht damit, daß alle Prozesse bestmöglich rechtzeitig vor ihrem nächsten Zyklusbeginn (=Termin) bedient werden. Eine spürbar höhere Wahrscheinlichkeit für die Einhaltung solcher Termine erzielt man unter Umständen durch Anwendung einer terminorientierten Abfertigungsstrategie anstelle der Verwendung statischer Prioritäten (siehe III).

(II) Gleichberechtigte und deshalb gleichwichtige Rechenprozesse sollten dieselbe statische unterbrechende Priorität haben. Das spart Unterbrechungsaufwand. Die so entstandene Gruppe von Prozessen kann entweder

a) mehrere Typen von Rechenprozessen mit unterschiedlichen Bediendauerverteilungen bzw. Periodendauern und Ankunftsraten oder
b) nur einen Typ enthalten.

Streuen im Fall b) die Bediendauern stärker als eine negative exponentielle Verteilung, dann sollten interne Prioritäten verwendet werden. Die mittlere Wartezeit und die Wartezeitverteilung aller Prozesse der Gruppe werden dann deutlich verbessert. Bei kleinerer Streuung ist die Abfertigung in der Reihenfolge des Eintreffens (FCFS) optimal.
Streuen im Fall a) die Bediendauern aller Typen von Prozessen nicht stärker als bei einer negativ exponentiellen Verteilung, dann ist die Zuteilung statischer nicht-unterbrechender (bei periodischen Prozessen unterbrechender) Prioritäten für unterschiedliche Typen nach der in (I) genannten Regel optimal. Je Typ sollte die Strategie FCFS angewandt werden. Bei stärkerer Streuung einzelner Typen und einer spürbaren Prozessorbelastung durch solche Prozesse sollten interne Prioritäten verwendet werden.

(III) Termine lassen sich (innerhalb einer Gruppe gleichberechtigter Prozesse verschiedenen Typs) durch eine einfache Strategie unter Anwendung dynamischer (d.h. wartezeitabhängiger) Prioritäten besonders gut einhalten. Das gilt sowohl für periodisch aktivierte wie durch spontane Ereignisse aktivierte Prozesse. Ein Typ umfaßt alle Prozesse, deren Termine gleich gut eingehalten werden sollen. Unterschiedlich wichtige Gruppen erhalten unterschiedliche statische unterbrechende Prioritäten. Die Termineinhaltungswahrscheinlichkeit nimmt dann mit steigender Priorität zu. Terminorientierte Abfertigung spart gegenüber optimalen statischen unterbrechenden Prioritäten (vgl. I) spürbar Unterbrechungen ein. Bei periodisch aktivierten Prozessen und terminorientierter Abfertigung darf eventuell eine deutlich höhere Prozessorbelastung zugelassen werden als bei Anwendung optimal nach (I) zugeteilter statischer unterbrechender Prioritäten.

(IV) Interne Prioritäten arbeiten mit Zeitscheiben, deren Länge oft von der erwarteten Restbediendauer eines Prozesses abhängig ist. Den Rechnerkern erhält innerhalb einer Gruppe gleichberechtigter Prozesse derjenige für die Dauer einer Zeitscheibe, der baldmöglichst fertigstellbar erscheint. Es sind also (statistische) Vorkenntnisse über die Bediendauern einzelner Rechenprozesse nötig. Interne Prioritäten sind dann vorteilhaft, wenn die Bediendauern stark streuen.

(V) Transiente (nicht zentralspeicherresidente) Prozesse sind so zu programmieren, daß während ihrer Bearbeitung möglichst wenige Hintergrundtransporte wegen *overlay*, *segmenting* oder *paging* auftreten. Das gilt für Grund- und Anwendungssoftware gleichermaßen. Zusammengehörige Befehle und Daten gehören in dieselbe Transporteinheit (segment, page). Transportierte Blöcke sollten so groß sein, daß die Übertragungszeit der Daten wenigstens 20% der gesamten Transport-Bearbeitungsdauer (=mittlere Positionier- plus Zugriffs- plus Übertragungszeit) ausmacht. Ausnahmen sollten begründbar sein. Transport-Bearbeitungszeiten sind
* Beiträge zur Wartezeit des verursachenden Prozesses,
* meist vollständige Beiträge zur Wartezeit aller anderen gleichzeitig wartenden Prozesse, denn die Überlappung von Transport durch Rechenarbeit ist bei Prozeßrechnern nach aller Erfahrung nur sehr gering.

Falls möglich, sollte mit kleinem Multiprogramminggrad (1 oder 2) gearbeitet werden, um möglichst große working-sets der geladenen Prozesse zu erzielen, um damit die Transportrate zu senken und um die gemeinsame mittlere Wartezeit aller transienten Prozesse zu verkleinern.
Zusätzlicher Zentralspeicherplatz vermindert oft die Transportrate und damit die Wartezeiten erheblich. Anstelle einer kostspieligen und raffinierten Organisation mit hohem Multiprogramminggrad und kleinen working-sets und dem dabei hohen Risiko falscher Parametereinstellungen sollte man die *billigere* und *transparentere*

Lösung wählen: Ein- (oder Zwei-)programmbetrieb mit einem großen Zentralspeicher.

2. Definitionen und Abkürzungen

Anforderung: durch Takt (interner Wecker) oder Dateneinfall (als Folge eines Interrupts oder Systemaufrufs, SVC) verursachter Rechenprozeß;

Ankunftsprozeß der Anforderungen: Gesetzmäßigkeit für den zeitlichen Abstand zwischen aufeinanderfolgenden Anforderungen (z. B. *konstant, zufällig*);

Poisson-Ankunftsprozeß: negativ exponentiell verteilte Ankunftsabstände;

Bediendauer: gesamte Belegungsdauer des Prozessors (beispielsweise des Rechnerkerns oder des Kanals) durch eine Anforderung;

Verweildauer: Zeit zwischen Ankunfts- und Fertigstellungszeitpunkt einer Anforderung;

Wartezeit: Verweildauer minus Bediendauer;

Verteilungsfunktion (VF): gibt Auskunft über die wahrscheinliche Dauer einer betrachteten Variablen (wie etwa Bediendauer, Wartezeit, usw.);

Mittelwert: Erwartungswert (=1. Moment) einer VF;

Streuung (Varianz): 2. Moment minus (Mittelwert)^2

Variationskoeffizient einer VF: $\sqrt{\text{Streuung}}$/Mittelwert; Maß zum Vergleich von VFn.

Der Variationskoeffizient ist

- 0 für konstante,
- 1 für negativ exponentielle,
- >1 für „stark streuende" (z. B. hyperexponentielle)

} Bediendauer-VFn

Typ einer Anforderung: Anforderungen desselben Typs entstammen demselben Ankunftsprozeß und haben dieselbe Bediendauer-VF;

charakteristische Größe: Größe zur Kennzeichnung des Verhaltens oder der Leistung eines Realzeitrechners, siehe „Betriebsziel";

Priorität: eine Zahl, welche die Wichtigkeit einer Anforderung für die Bearbeitung durch den Prozessor angibt; kleine Zahl bedeutet hohe Priorität (Wichtigkeit);

Priorität, statische: einmalig beim Auftreten einer Anforderung für diese festgelegte Priorität;

Priorität, dynamische[1]: von extern mitgelieferten Bedingungen, z. B. für die maximale individuelle Wartezeit einer Anforderung, abhängige, veränderliche Priorität;

Priorität, interne: von internen Bedingungen, z. B. der erwarteten oder bekannten Rest-Bediendauer, abhängige, veränderliche Priorität;

Priorität, unterbrechende: eine Anforderung mit höherer Priorität hat unterbrechende Priorität gegenüber einer anderen, wenn sie sofort unter Verdrängung der anderen den Prozessor erhält;

Priorität, nichtunterbrechende: eine in Arbeit befindliche Anforderung kann nicht mehr, unabhängig von den Prioritäten anderer wartender Anforderungen, unterbrochen werden;

Prioritätsebene: Warteschlange für alle Anforderungen gleicher statischer unterbrechender Priorität (meist hardware-unterstützt).
Anforderungen innerhalb einer Prioritätsebene können untereinander unterschiedliche
- statische nicht-unterbrechende,
- dynamische unterbrechende oder nichtunterbrechende,
- interne

Prioritäten haben;

Gruppe von Anforderungen: ein oder mehrere Typen von Anforderungen mit bezüglich der charakteristischen Größen gleichen Optimierungszielen. Beispielsweise ist nur die gemeinsame Wartezeit oder die typspezifische Termin-Einhaltung interessant;

Strategie: Anweisung, die den Einsatz eines bestimmten Betriebsmittels (z. B. Prozessor), insbesondere die Auswahl einer aus evtl. mehreren wartenden Anforderungen zur Bearbeitung, steuert;

Betriebsziel: charakteristische Größe, die optimiert werden sollte, z. B. Wartezeit (Mittelwert, Streuung), Termin für Bearbeitungsbeginn oder -ende, Zahl von Unterbrechungen, individuell abgestufte Wartezeit je nach Typ einer Anforderung usw.

Häufig verwendete Abkürzungen

β_i Mittel-, Erwartungswert der Bediendauer von Anforderungen des Typs i;

$\beta_{\leq k} = \lambda_{\leq k}^{-1} \sum_{i=1}^{k} \lambda_i \beta_i$ Erwartungswert der gewichteten Bediendauer-Verteilungen von Anforderungen vom Typ 1 bis k;

C Variationskoeffizient;

λ_j Ankunftsrate des Ankunftsprozesses für Anforderungen vom Typ i;

$\lambda_{\leq k} = \sum_{i=1}^{k} \lambda_i$ Summierte Ankunftsraten für Anforderungen vom Typ 1 bis k;

ν_i Wichtigkeit (=freiwillige Wartezeit, die möglichst nicht überschritten werden sollte) einer Anforderung vom Typ i;

$P\{a \leq t\} = P(\leq t)$ Kleinergleich-Verteilungsfunktion, Wahrscheinlichkeit, daß eine Zufallsgröße den Wert $a \leq t$ annimmt;

ϱ_i Belastung des Prozessors (Angebot) durch Anforderungen vom Typ i

$\varrho_{\leq k} = \sum_{i=1}^{k} \varrho_i$ Gesamtangebot von Anforderungen vom Typ 1 bis k;

V mittlere Verweildauer (=Verweilzeit);

VF, VFn Verteilungsfunktion, -en;

$W(\leq t)$ Wartezeit-VF

W mittlere Wartezeit ($=V-\beta$) (Erwartungswert);

$w(t)$ tatsächliche aufgelaufene Wartezeit;

$W_{\leq N} = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_{\leq N}} W_i$ gemeinsame mittlere Wartezeit einer Gruppe aus N Typen von Anforderungen.

Strategien

FCFS first come first serve; Ankunftsreihenfolge einhalten;

SERPT shortest expected remaining processing time first; Anforderung mit kürzester erwarteter Restbediendauer zuerst bearbeiten;

RU relative urgency; Anforderungen entsprechend mitgebrachter Terminwünsche bearbeiten.

---

[1]) Der Begriff dynamische Priorität wird hier nicht, wie beim Betriebssystem MARTOS der AEG 80 Familie üblich, als die Möglichkeit verstanden, die statische Priorität eines Prozesses durch einen anderen, z. B. den aufrufenden, Prozeß zu ändern. Da die Priorität eines Prozesses dadurch während seiner Lebensdauer in der Regel nur einmal geändert wird, entspricht das im Rahmen unserer Modelle einer statischen Prioritätsvergabe, wobei – je nach Auftraggeber – die eine oder andere statische Prioritätsnummer gilt. Der zur Änderung der Priorität vorhandene Mechanismus könnte, wenn er effektiv genug wäre, auch bei den hier beschriebenen dynamischen und internen Prioritäten eingesetzt werden.

3. Modelle und Optimierungsziele

Die Bilder 1 und 2 zeigen Rechner-Modelle, wie sie zur Analyse des kritischen Echtzeitbetriebs geeignet sind. In Bild 1 treffen Anforderungen mit der Ankunftsrate λ ein und werden in einer gemeinsamen Warteschlange verwaltet. Optimierungsziel ist die Wartezeit, insbesondere ihr Mittelwert und ihre Streuung. In Bild 2 gibt es N Typen von Anforderungen und dementsprechend N verschiedene Warteschlangen. Optimierungsziele können sein

a) die uneingeschränkte Bevorzugung von Anforderungen des Typs i gegenüber solchem vom Typ $j>i$,
b) die wartezeitgünstige Abfertigung einer bestimmten Gruppe (mit einem oder mehreren Typen) von Anforderungen,
c) die bestmögliche Garantie von Terminen

oder mehrere der genannten Ziele nebeneinander. Bild 3 zeigt ein Rechner-Modell, das für die Analyse des unkritischen Echtzeitbetriebs geeignet ist: Anforderungen treffen mit der Rate λ ein und haben zur Folge, daß nach einem Datentransport vom Hintergrund- in den Zentralspeicher der Rechnerkern für sie arbeiten kann. Die Rechnerkern-Arbeit muß unter Umständen mehrmals vorübergehend unterbrochen werden, um weitere Daten zwischen Hintergrund- und Zentralspeicher (oder zurück) zu transportieren. Dieses Modell entspricht dem von Bild 1 mit dem Unterschied, daß Transporte zwischen Hintergrund- und Zentralspeicher einbezogen werden. Bei Auftreten eines Transportwunsches wird der im Modell eingezeichnete Rückkopplungsweg benutzt. Wollte man, entsprechend dem Modell von Bild 2, verschiedene Typen von Anforderungen unterscheiden und dabei Transporte berücksichtigen, dann müßte man die einzelnen Warteschlangen des Modells von Bild 3 durch mehrere parallel angeordnete Warteschlangen (je eine pro Typ) ersetzen (vgl. Bild 4).

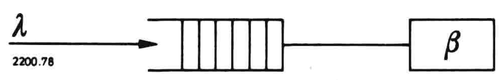


Bild 1: Modell mit einer Warteschlange

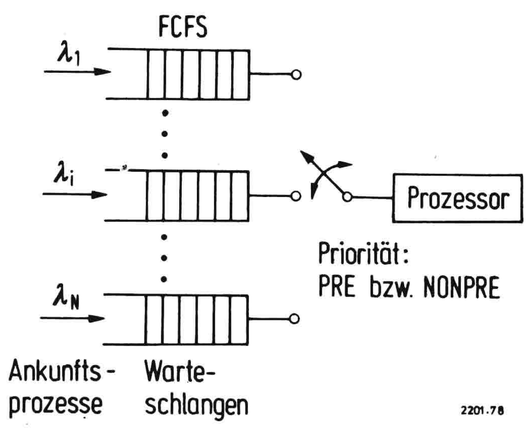


Bild 2: Verkehrstheoretisches Modell eines Realzeitrechners im kritischen Echtzeitbetrieb

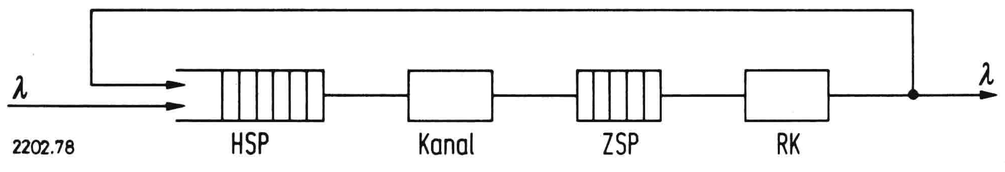


Bild 3: Warteschlangen-Netz-Modell eines Realzeitrechners im unkritischen Echtzeitbetrieb mit Ankunfts- und Fertigstellungsrate λ (Anforderungen pro Zeiteinheit), unbegrenzt großem Hintergrundspeicher HSP, einem Transportkanal, Zentralspeicher ZSP mit n Plätzen und einem Rechnerkern RK

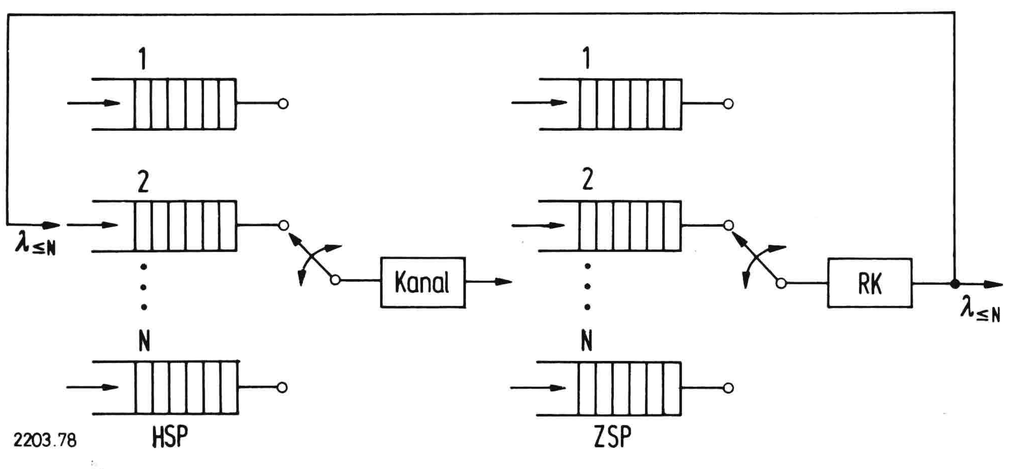


Bild 4: Warteschlangen-Netz entsprechend Bild 3, aber mit N unterschiedlichen Typen von Anforderungen. HSP = Hintergrundspeicher, ZSP = Zentralspeicher, RK = Rechnerkern

Optimierungsziele für unkritische Echtzeitanforderungen können sowohl die oben als auch zusätzlich die nachfolgend genannten sein:

d) Unterschiedliche (gestaffelte) mittlere Wartezeiten entsprechend dem Typ einer Anforderung (nicht rigorose Bevorzugung wie bei a).
e) Proportionalität zwischen mittlerer Wartezeit und gesamter mittlerer Bediendauer (= Rechen- plus Transportzeit) einer Anforderung

und so weiter. Diese Optimierungsziele und die Strategien zu ihrer Erreichung werden hier nicht besprochen.

Die Modelle für unkritische Echtzeitanforderungen sind Erweiterungen der einfacheren Modelle, wie sie bei kritischen Echtzeitanforderungen üblich sind. Bei gleichen Optimierungszielen ergibt sich unabhängig von der Art der Echtzeitanforderungen (kritisch/unkritisch) jeweils dieselbe Strategie als optimal bzw. besonders günstig. Tabelle 1 gibt eine Übersicht über die hier behandelten Optimierungsziele, Modelle und Ergebnisse.

Für kritische Echtzeitanforderungen, also die Modelle der Bilder 1 und 2, lassen sich für unterschiedlichste Belastungsfälle und Optimierungsziele die jeweils optimalen Strategien angeben und meistens auch zugehörige charakteristische Größen berechnen. In Abschnitt 4 werden entsprechende Beispiele behandelt.

Bezieht man Hintergrundtransporte in die Modellierung ein, dann sind optimale Strategien nur für Sonderfälle angeb- und berechenbar. Nahezu optimale Strategien sind jedoch für fast alle Belastungsfälle bekannt[2]). Die Berechnung von Zahlenwerten für die Optimierungsziele gelingt auch dann nur in Sonderfällen. Abschnitt 5 bringt Beispiele dazu.

4. Optimale Strategien für kritische Echtzeitanforderungen und verschiedene Optimierungsziele

Wir geben jetzt zu den einzelnen Optimierungszielen optimale Strategien sowie Beispiele für die dabei erzielten Zahlenwerte an. Das in Bild 1 gezeigte Modell mit nur einem Typ von Anforderungen ist als Grenzfall im Modell nach Bild 2 für $N=1$ enthalten. Wir behandeln darum das Modell nach Bild 2 mit $N \geqslant 1$.

4.1 Optimierungsziel: Uneingeschränkte Bevorzugung von Anforderungen des Typs i gegenüber anderen vom Typ j>i

Optimal: statische unterbrechende Prioritäten. Je wichtiger die Anforderungen sind, um so höher muß die statische Priorität sein.

Bemerkungen: Um den mit Unterbrechungen verbundenen Verwaltungsaufwand gering zu halten und die Übersichtlichkeit zu erhalten, sollte die Anzahl der unterbrechenden Prioritäten klein sein. Etwa gleichwichtige Anforderungen (und die zugehörigen Rechenprozesse) sollten derselben Prioritätsebene zugeordnet werden, also in einer Gruppe zusammengefaßt werden.

4.1.1 Ergebnisse für den ereignisgesteuerten Betrieb

Im ereignisgesteuerten Betrieb treffen Anforderungen zu nicht vorher bekannten Zeitpunkten ein. Das Produkt aus gesamter Ankunftsrate $\lambda_{\leq N}$ und gemeinsamer mittlerer Bediendauer $\beta_{\leq N}$ aller Anforderungen heißt Gesamtangebot. Wir betrachten ein

Beispiel 4.1

Modell nach Bild 2, N=4, gleiche Angebote für alle Typen von Anforderungen $\varrho_i = \varrho_{\leq N}/N$, Poisson-Ankunftsprozesse, konstante Bediendauern der Dauer β_i. Bei der Zuordnung der β_i zu den Prioritäten i unterscheiden wir noch zwei Fälle:

4.1 a) $\beta_1=1$, $\beta_2=3$, $\beta_3=5$, $\beta_4=7$,
4.1 b) $\beta_1=7$, $\beta_2=5$, $\beta_3=3$, $\beta_4=1$.

Dabei ist sowohl für a) wie b) die gemeinsame mittlere Bediendauer $\beta_{\leq N}=2{,}4$.

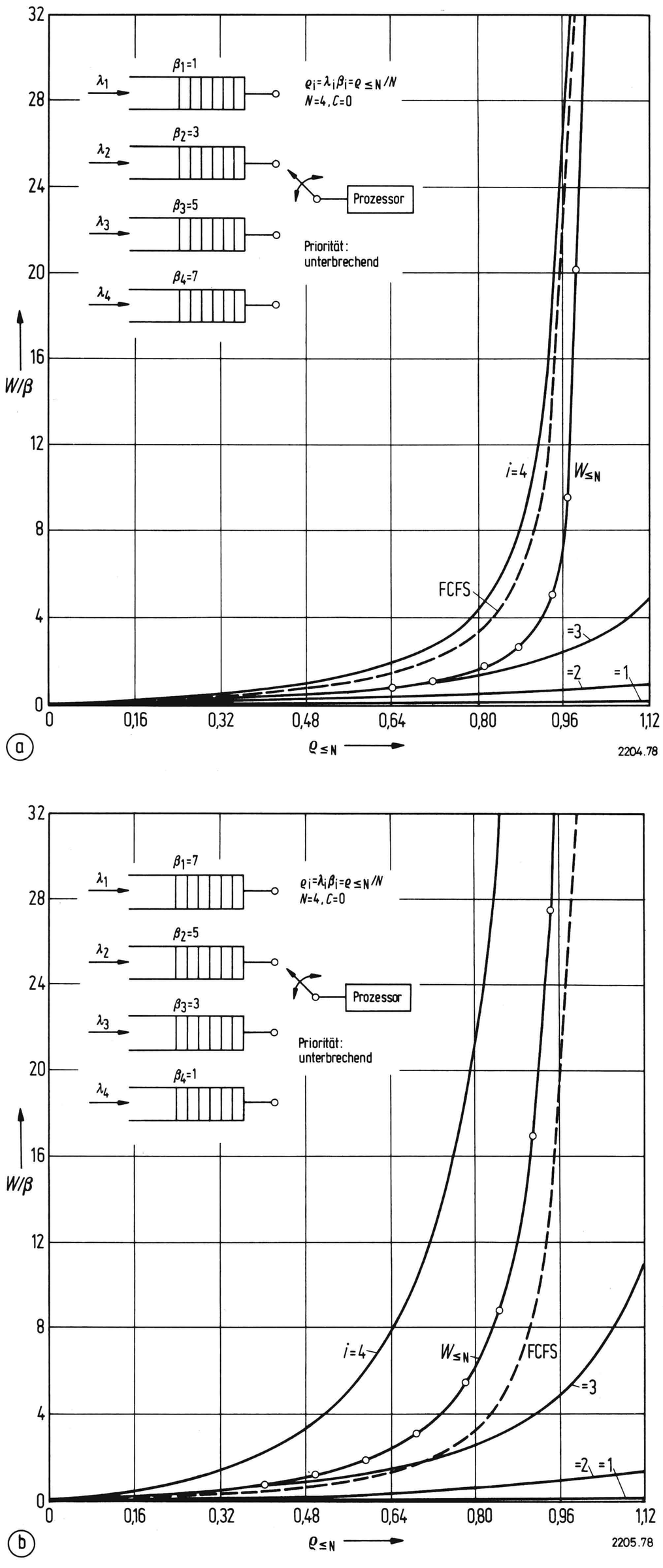
---

[2]) Eine Eigenart dieser Modelle ist, daß optimale Strategien häufig zustandsabhängig sein müssen. Vernachlässigt man das, so kommt man zu stationär optimalen („nahezu optimalen") Strategien.

Wir unterstellen, daß Anforderungen desselben Typs *i* nacheinander (in der Reihenfolge des Eintreffens = FCFS) abgearbeitet werden. Anforderungen unterschiedlichen Typs sollen unterschiedliche statische Prioritäten haben.

Man kann die bezogene mittlere Wartezeit W_i/β_i der Anforderungen des Typs *i* bei unterbrechenden und nicht-unterbrechenden Prioritäten geschlossen berechnen und erhält für dieses Beispiel 4.1 a) bzw. 4.1 b) für *unterbrechende* Prioritäten die Kurven in den Bildern 5 a bzw. 5 b. An der Abszisse steht das Gesamtangebot $\varrho_{\leq N}$, an der Ordinate die auf die mittlere Bediendauer bezogene mittlere Wartezeit W/β. Die mit FCFS bezeichneten Kurven sind identisch; sie ergeben sich beim Verzicht auf statische Prioritäten, wobei das Modell nach Bild 2 in das nach Bild 1 degeneriert.

Man sieht durch Vergleich der Bilder 5 a und 5 b, daß für das gleiche Belastungsbeispiel (aber unterschiedliche Prioritätsvergabe), abhängig von der statischen Priorität *i* die bezogene mittlere Wartezeit einer Anforderung sehr unterschiedlich sein kann. So wird bei $\varrho_{\leq N} = 0{,}8$ die bezogene mittlere Wartezeit der Anforde-



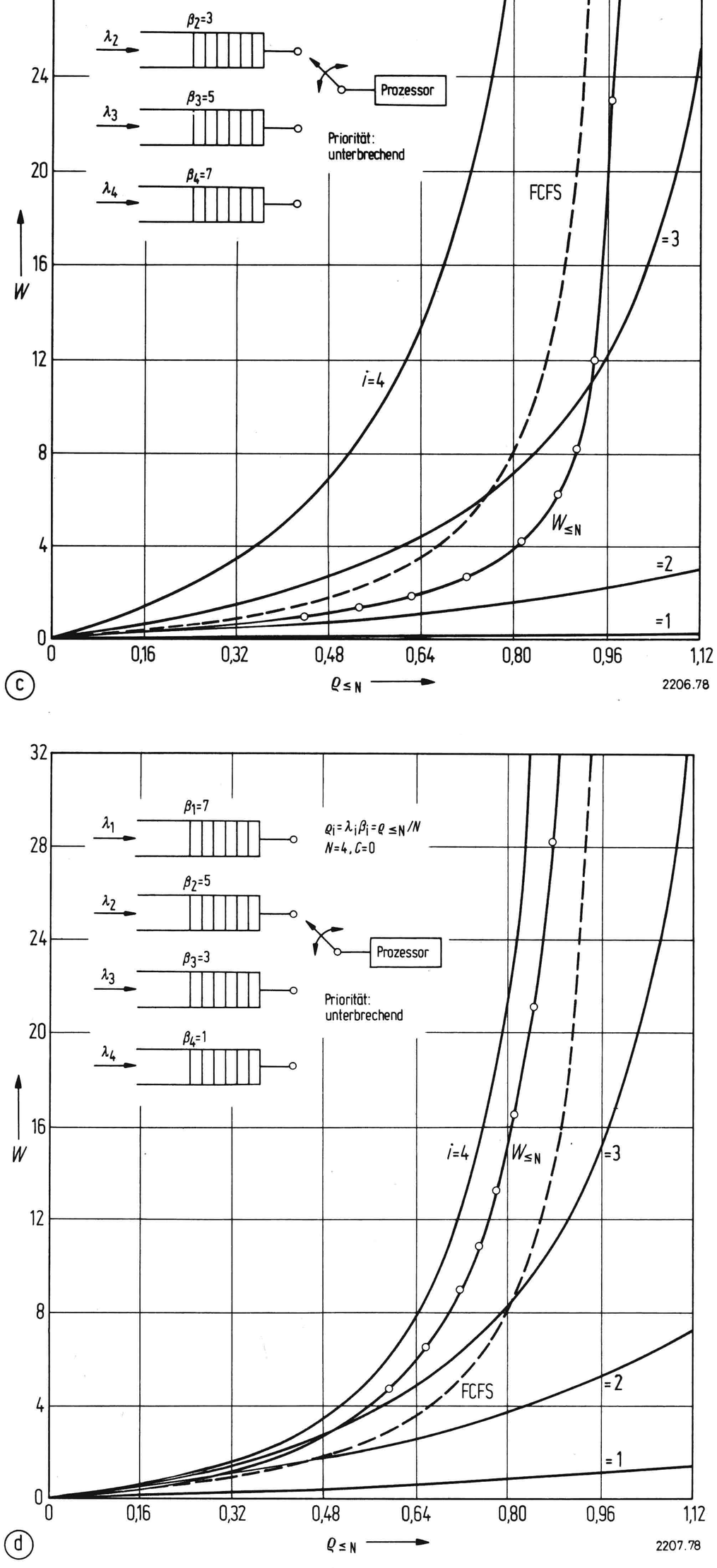


Bild 5: Bezogene mittlere Wartezeit W/β über dem Angebot bei unterbrechenden Prioritäten.
Für die mit $i=1$ bis 4 bezifferten Kurven ist $W/\beta = W_i/\beta_i$. Für die Kurven FCFS bzw. $W_{\leq N}$ ist $W/\beta = W_{FCFS}/\beta_{\leq N}$ bzw. $W_{\leq N}/\beta_{\leq N}$.
Modell und Belastungsbeispiele sind im Bild eingetragen. An den Ordinaten steht die mittlere Wartezeit anstelle der bezogenen mittleren Wartezeit wie in den Bildern 5 a) und b). Es entsprechen sich die Bilder 5 a) und c) sowie 5 b) und d).

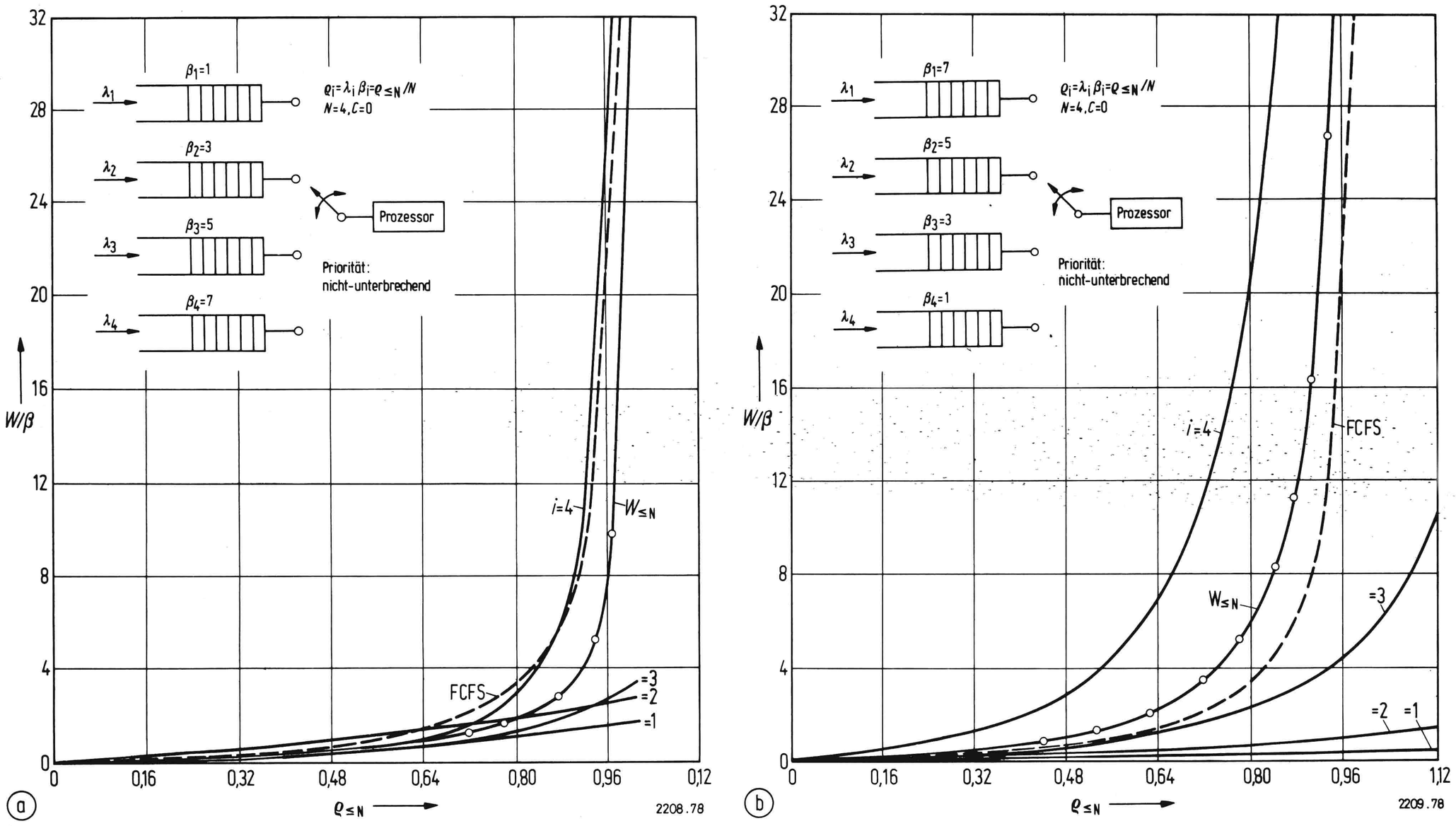


Bild 6: Bezogene mittlere Wartezeit W/β (Bilder a und b) bzw. mittlere Wartezeit W (Bild c) über dem Gesamtangebot bei nicht-unterbrechenden Prioritäten. Modell und Belastungsbeispiele sind in den Bildern eingetragen. Bild c) zeigt die Ergebnisse für die in a) und b) dargestellten Belastungsbeispiele gemeinsam. *Nur* die gemeinsame mittlere Wartezeit $W_{\leq N}$ ist Belastungsbeispiel-abhängig und ist dort deshalb mit a) bzw. b) gekennzeichnet.

rungen mit dem Bediendauer-Erwartungswert $\beta_1 = 1$ im Bild 5a (wo sie die höchste Priorität $i=1$ haben) $W_1/\beta_1 \approx 0$. Dagegen erreicht man für dieselben Anforderungen, wenn sie die niedrigste Priorität $i=4$ erhalten, nur $W_4/\beta_4 = 20$ (vgl. Bild 5b). Die Normierung der mittleren Wartezeit W_i auf die zugehörige mittlere Bediendauer β_i verdeckt die Relationen der mittleren Wartezeiten unterschiedlicher Typen von Anforderungen zueinander, wenn sie verschiedene Bediendauer-Erwartungswerte β_i haben. Die Normierung führt aber zur interessanteren Aussage, denn die mittlere Wartezeit muß meistens in Relation zur mittleren Bediendauer gesehen werden. Ohne diese Normierung erhält man Ergebnisse, wie sie in den Bildern 5c und 5d dargestellt werden.

Für *nicht-unterbrechende* Prioritäten erhält man für dasselbe Belastungsbeispiel die in den Bildern 6a und 6b gezeigten Ergebnisse: Vergleichbare Ergebnisse von *unterbrechenden* Prioritäten zeigen die Bilder 5a und 5b. Spektakuläre Unterschiede sind nicht feststellbar. Die Streuung der mittleren Wartezeiten W_i ist bei unterbrechenden Prioritäten (Bilder 5a, b) größer als bei nicht-unterbrechenden (Bilder 6a, b). Betrachtet man die mittleren Wartezeiten anstelle der bezogenen mittleren Wartezeiten, dann stellt man fest (Bild 6c), daß sie nicht von der Prioritätsvergabe abhängen: Für die Beispiele 4.1 a) und 4.1 b) ergibt sich dasselbe. Das liegt allein an der Annahme gleicher Angebote ϱ_i für alle Typen von Anforderungen. Dagegen sind die gemeinsamen mittleren Wartezeiten $W_{\leq N}$ von der Prioritätsvergabe abhängig.

Man kann die Wartezeit-Verteilungsfunktionen $P(w_i \leq t)$, nämlich die Wahrscheinlichkeit, daß eine Wartezeit w_i einer Anforderung vom Typ i kleiner oder gleich einer Zeit t sein wird, näherungsweise berechnen. Bild 7 zeigt Simulationsergebnisse für den Fall unterbrechender Prioritäten. Beispielsweise wird eine Wartezeit $t=20$ Zeiteinheiten für Anforderungen mit dem Bediendauer-Erwartungswert $\beta=5$ bei einer Prioritätszuteilung nach Beispiel 4.1 a) mit einer Wahrscheinlichkeit von $P(\leq t)=0{,}96$ eingehalten oder unterschritten (siehe gestrichelte Kurve, $i=3$).

Bei $t=0$ kann man die Wahrscheinlichkeit ablesen, daß eine Anforderung vom Typ i nicht warten muß. Bei unterbrechenden Prioritäten unterscheidet man Warten auf Bearbeitungsbeginn und Warten wegen Unterbrechung durch wichtigere Anforderungen. Beides zusammen ergibt die Wartezeit.

Die Ergebnisse zeigen, daß unterschiedliche statische Prioritätszuordnungen für den gleichen Belastungsfall zu deutlich unterschiedlichen Ergebnissen für die charakteristischen Größen mittlere Wartezeit und Wartezeit-VF führen. Man sollte beachten, daß die *gemeinsame mittlere Wartezeit* $W_{\leq N}$ aller Anforderungen oft dann minimal oder besonders klein wird, wenn man die Prioritäten nach folgender Regel so zuteilt, daß

$$\beta_1 \leq \beta_2 \leq \ldots \leq \beta_N \qquad (1)$$

erfüllt ist. Das heißt, je größer der Bediendauer-Erwartungswert, um so kleiner sollte die Priorität sein. Bei unterbrechenden Prioritäten gilt diese Regel exakt oder mit guter Näherung *nur*, wenn jede Bediendauer-VF der verschiedenen Typen von Anforderungen ebenso oder weniger streut als eine negativ exponentielle Verteilung. Dagegen gilt die Regel bei nicht-unterbrechenden Prioritäten uneingeschränkt.

Im Beispiel 4.1 a) wird diese Regel beachtet. Man sieht durch Vergleich der Bilder 5a und 5b bzw. 6a und 6b, daß die bezogene gemeinsame mittlere Wartezeit (Kurve $W_{\leq N}$ in Bild 5a bzw. 6a deutlich vorteilhafter verläuft als in Bild 5b bzw. 6b. (Dazu betrachte man den Abstand zur Kurve FCFS, die in beiden Bildern identisch verläuft.) Die Wartezeit-VFn bei nicht-unterbrechenden Prioritäten für dasselbe Belastungsbeispiel (4.1) zeigt Bild 8. Im Unterschied zu Bild 7 müssen jetzt alle Anforderungen mit der gleichen Wahrscheinlichkeit $P(\leq t) = 1 - \varrho_{\leq N}$ nicht warten.

Aus den in den Bildern 7 und 8 dargestellten Ergebnissen für die Wartezeit-VFn der Anforderungen ersieht man, daß je nach Typ der Anforderung die Wahrscheinlichkeit für die Einhaltung bzw. Unterschreitung einer festgegebenen Zeit t *sehr* unterschied-

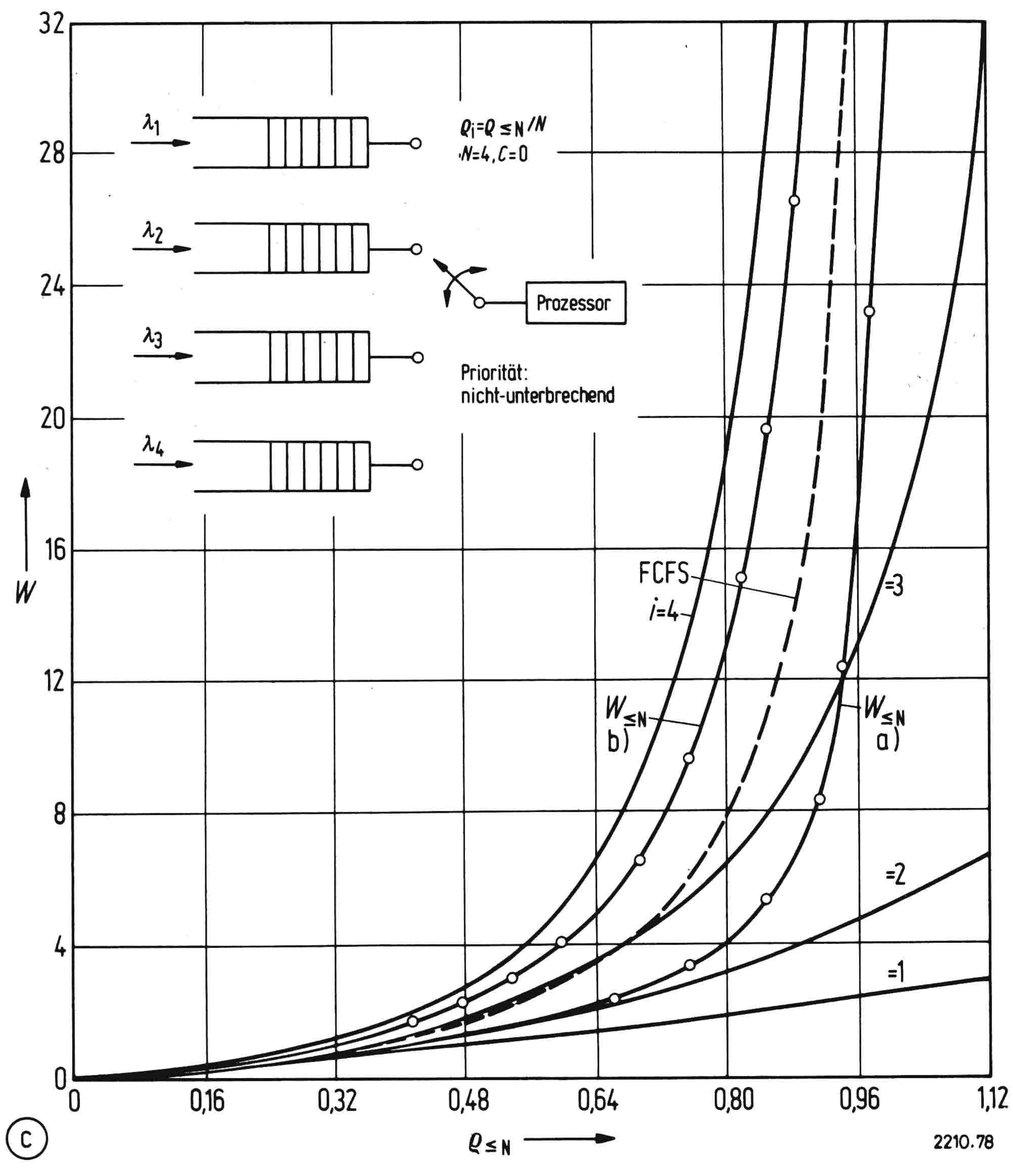


Bild 6c): Mittlere Wartezeit über dem Gesamtangebot bei nicht-unterbrechenden Prioritäten. Bei Voraussetzung gleicher Angebote ρ_i sind die mittleren Wartezeiten W_i beider in den Bildern 6a) und b) gezeigter Belastungsbeispiele unabhängig von der Prioritätszuordnung. Die gemeinsamen mittleren Wartezeiten sind jedoch verschieden; vgl. die Kurven $W_{\leq N}$ (Bild 6a) bzw. $W_{\leq N}$ (Bild 6b).

lich ausfällt. Anforderungen mit hoher Priorität erfahren viel wahrscheinlicher kleine Wartezeiten als solche mit geringer Priorität.

Der Charakter der Ergebnisse ändert sich nicht, wenn anstelle vorausbekannter Bediendauern, wie bisher im Beispiel 4.1 unterstellt, durch eine Wahrscheinlichkeits-VF gegebene Bediendauern (also nur statistische Kenntnisse) vorliegen. Bei Anwendung der Regel (1) werden die Wartezeit-VFn aller Typen von Anforderungen innerhalb der Gruppe besonders günstig. Beispielsweise wird die gemittelte Überschreitungswahrscheinlichkeit

$$\ddot{U} = \sum_{i=1}^{N} (\lambda_i / \lambda_{\leq N})\, P(w_i > t_i) \qquad (2)$$

die man aus den gewichteten und summierten Überschreitungswahrscheinlichkeiten $P(w_i > t) = 1 - P(w_i \leq t)$ berechnet, besonders klein. Dabei ist t_i irgendeine *feste* Wartezeit, die für die Anforderungen des Typs i betrachtet wird, beispielsweise darf auch sein $t_1 = t_2 = \ldots = t_N$.

Für das Beispiel 4.1 und die (hier beispielhaft gewählten) Wartezeiten $t_1 = 15$, $t_2 = 25$, $t_3 = 35$, $t_4 = 45$ berechnet man mit Hilfe der Ergebnisse aus Bild 7 für unterbrechende Prioritäten und eine Prioritätenzuordnung nach der Regel (1): $\ddot{U}_{opt} = 0{,}012$ und bei ungünstigster Prioritätenzuordnung: $\ddot{U}_{pess} = 0{,}056$. Die Überschreitungswahrscheinlichkeit ist hier bei praktisch optimaler Prioritätenzuteilung viermal kleiner als bei ungünstigster Zuteilung.

4.1.2 Ergebnisse für den taktgesteuerten Betrieb

Im taktgesteuerten Betrieb treffen Anforderungen periodisch zu vorher bekannten Zeitpunkten ein, wobei man die Periodendauer T_i von Anforderungen des Typs i und die Bediendauer β_i pro

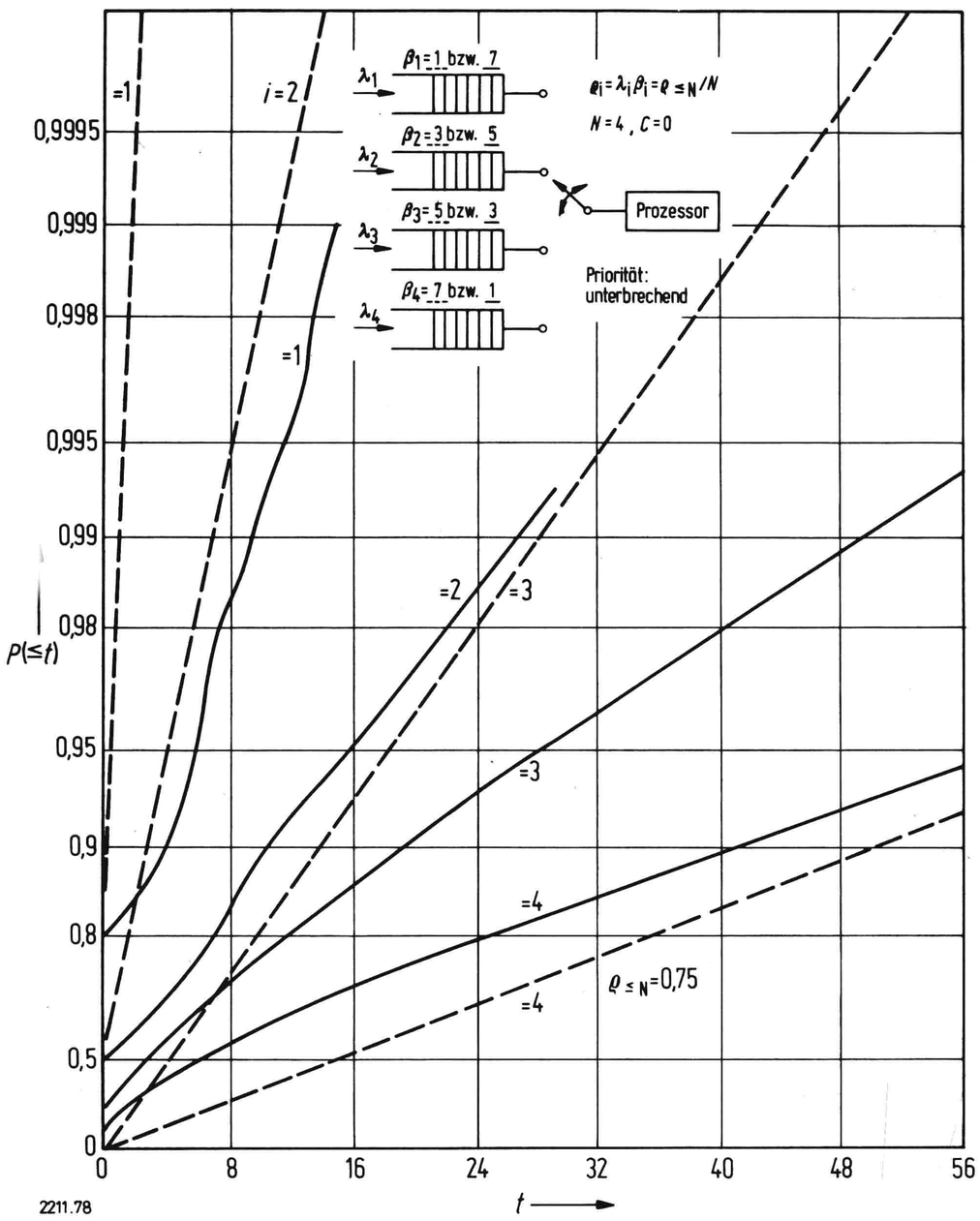


Bild 7: Unterbrechende statische Prioritäten.
Wartezeitverteilungsfunktionen bei konstantem Angebot $\rho_{\leq N} = 0{,}75$. Der im Belastungsbeispiel (siehe Parameter des Warteschlangenmodells im Bild) zuerst genannte Bediendauer-Erwartungswert β_i gilt für die gestrichelten Kurven, der zweitgenannte für die durchgezogenen. Alle Kurven wurden durch Simulation gewonnen, sind jedoch auch näherungsweise berechenbar.

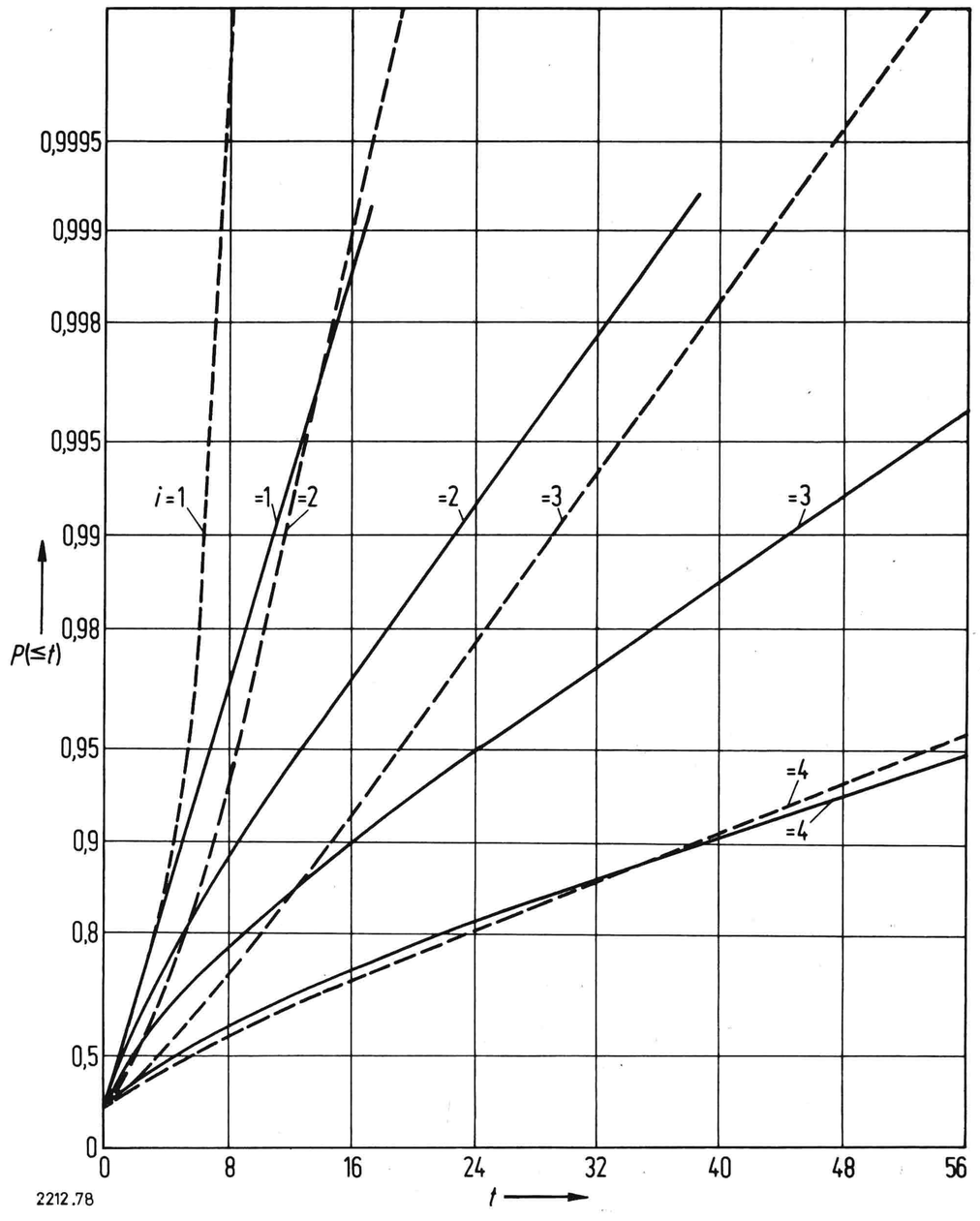


Bild 8: Nicht-unterbrechende statische Prioritäten.
Wartezeitverteilungsfunktionen bei konstantem Angebot $\rho_{\leq N} = 0{,}75$. Der im Belastungsbeispiel (siehe Parameter des Warteschlangenmodells im Bild) zuerst genannte Bediendauer-Erwartungswert β_i gilt für die gestrichelten Kurven, die zweitgenannte für die durchgezogenen. Alle Kurven wurden durch Simulation gewonnen, sind jedoch auch näherungsweise berechenbar.

Periode kennt. Entsprechend dem Modell nach Bild 2 teilt man Anforderungen mit unterschiedlicher Periodendauer unterschiedliche Prioritäten zu.
Die bestmögliche Abarbeitung von N Typen von Anforderungen mit je individueller Periodendauer erreicht man (im Rahmen der Anwendung unterbrechender statischer Prioritäten), indem man die Priorität mit zunehmender Periodendauer abnehmen läßt. Die Prioritäten sind optimal zugeordnet, wenn erfüllt ist

$$T_1 < T_2 < T_3 < \ldots < T_N \qquad (3)$$

Sind die einzelnen Periodendauern keine ganzzahligen Vielfachen voneinander, dann darf das Gesamtangebot nur

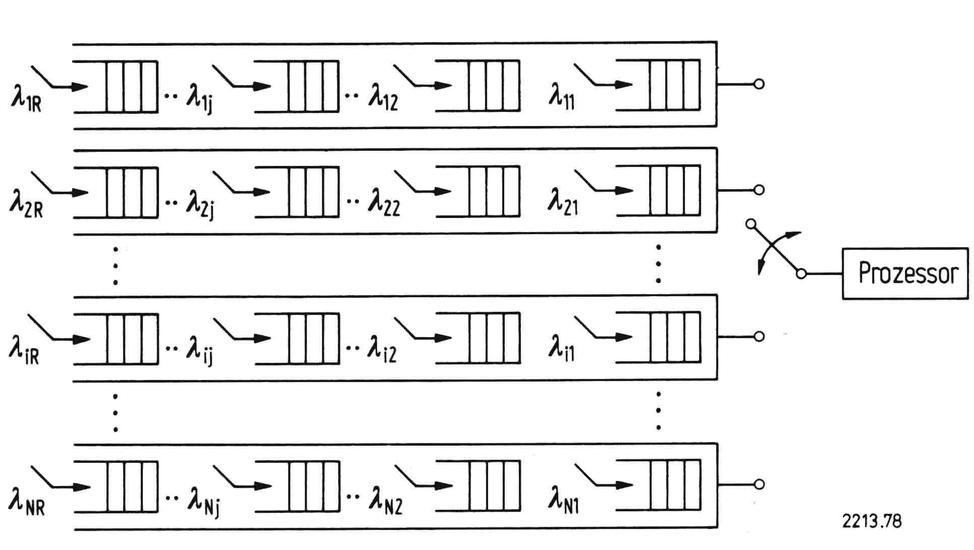
$$\varrho_{\leq N} \leq N(2^{1/N} - 1)$$

sein, damit alle Anforderungen rechtzeitig vor Beginn des nächsten Zyklus (=Termin) fertiggestellt werden können.
(Beispiele: $N=4$ führt auf $\varrho_{\leq N}=0{,}76$, $N=10$ führt auf $\varrho_{\leq N}=0{,}72$.)
In Abschnitt 4.3.2 wird ein Beispiel besprochen (vgl. Bild 16a und 16b).
Sind die Periodendauern ganzzahlige Vielfache voneinander, dann darf der Prozessor sogar zu 100% ausgelastet werden, ohne daß ein Termin verpaßt wird.
Das Gesamtangebot ist aus der Summe der Periodenbruchteile bestimmbar, während denen Anforderungen vom Typ i den Prozessor belegen:

$$\varrho_{\leq N} = \sum_{i=1}^{N} \beta_i / T_i \qquad (4)$$

4.2 Optimierungsziel: Wartezeitgünstige Abfertigung einer bestimmten Gruppe von Anforderungen

Wir diskutieren hier nur den ereignisgesteuerten Betrieb. Eine Gruppe von Anforderungen besteht aus N Typen von gleichberechtigten Anforderungen ($N \leq 1$), so daß das Modell nach Bild 2 (aber ohne die Interpretation „Nummer der Warteschlange = statische Priorität") zutrifft. Neben der betrachteten Gruppe können *weitere* Gruppen von Anforderungen an den Realzeitrechner existieren. Unterschiedliche Gruppen werden in der Regel unterschiedliche statische Priorität haben. Ein solcher Fall soll zunächst betrachtet werden. Dazu unterstellen wir das Modell nach Bild 9 mit N statischen unterbrechenden Prioritäten (=Prioritätsebenen) und innerhalb jeder Ebene R Typen von Anforderungen. (Läßt man zu, daß einzelne Ankunftsraten λ_{ij} Null sind, dann kann die Zahl von tatsächlich vorhandenen Typen von Anforderungen je Ebene verschieden sein.) Unterstellt man, daß die Prioritätsebenen mit ihren zugehörigen Belastungsparametern aus irgendwelchen Erwägungen bereits optimal gewählt sind (vgl. Abschnitt 4.1), dann bleibt die Frage zu beantworten, wie die Anforderungen innerhalb *einer* Ebene – isoliert betrachtet – abzuarbeiten sind.



Modell 9: Modell mit N unterbrechenden Prioritäten (=Prioritätsebenen) und R Typen von Anforderungen je Ebene.

Für diese Überlegung reicht das Modell nach Bild 2 anstelle des Modells nach Bild 9 aus. Liegt nur ein Typ von Anforderungen innerhalb der betrachteten Gruppe vor, dann vereinfacht sich das Modell weiter zu dem nach Bild 1. Streuen die Bediendauern aller Typen von Anforderungen der betrachteten Gruppe jeweils nicht mehr als bei einer negativ exponentiellen VF, dann ist je Typ die Abarbeitungsreihenfolge FCFS und für konkurrierende Typen von Anforderungen die Vergabe statischer nicht-unterbrechender Prioritäten nach der Regel (1) vorteilhaft: die gemeinsame mittlere Wartezeit wird dann praktisch minimal. Die mittlere Wartezeit W_{ij} von Anforderungen der Ebene i und des Typs j dieser Ebene und ihre Streuung sind für das Modell nach Bild 9 geschlossen berechenbar, wenn man innerhalb der Ebene i nicht-unterbrechende Prioritäten anwendet [1].
Streuen die Bediendauern mindestens eines Typs von Anforderungen in der betrachteten Gruppe stärker als eine negativ exponentielle VF, und ist das von diesem Typ stammende Angebot nicht vernachlässigbar, dann sind interne Prioritäten zweckmäßiger als statische. Denn die gemeinsame mittlere Wartezeit und die Wartezeit-VF werden günstiger. Für die übrigen Typen von Anforderungen innerhalb der Gruppe (und damit Ebene), die wenig streuende Bediendauern haben, sind statische Prioritäten nach der Regel (1) anzuwenden, wobei Anforderungen gleichen Typs nacheinander bearbeitet werden. Von den Anforderungen mit stark streuender Bediendauer wird jeweils die zuerst bearbeitet, deren erwartete Restbediendauer am kürzesten ist. Ihre Priorität gegenüber den anderen Typen von Anforderungen ergibt sich aus der Regel (1), wobei für sie β_i gleich dem Restbediendauer-Erwartungswert einzusetzen ist. Diese Strategie ist relativ kompliziert, und Ergebnisse sind nur durch Simulation zu erhalten.
Wir betrachten zunächst einen vereinfachten Fall, bei dem jede Gruppe nur genau einen Typ von Anforderungen enthält. Er soll stark streuende Bediendauern haben. Das Modell nach Bild 9 geht dabei in das nach Bild 2 über. Es soll gezeigt werden, wie sehr sich die Anwendung interner Prioritäten innerhalb einer Prioritätsebene lohnen kann.

Beispiel 4.2
Modell nach Bild 2, $N=3$, unterbrechende statische Prioritäten, gleiches Angebot aller Typen von Anforderungen $\varrho_i = \varrho_{\leq N}/N$, gleiche stark streuende Bediendauer-VF für alle Anforderungen mit einem Variationskoeffizienten $C=3{,}5$ (eine negative exponentielle VF hat $C=1$), gleiche Erwartungswerte β_i.
Bearbeitet man die Anforderungen jeder Prioritätsebene i nach der Regel FCFS, dann berechnet man die in Bild 10 dargestellten, mit $i=1$ bis 3 bezifferten Kurven. Üblicherweise ist $\varrho_{\leq N}=1$.
Verwendet man in der Prioritätsebene i ein – bezüglich der mittleren Wartezeit – optimales Zeitscheibenverfahren [2], dann erhält man die gestrichelt gezeichneten Kurven. Beispielsweise könnte man in den Ebenen 1 und 3 die Regel FCFS und in der Ebene 2 die optimale Strategie anwenden. Dann erhielte man die durchgezogenen Kurven $i=1$ bzw. 3 und die gestrichelte Kurve $i=2$. Bild 11 zeigt Simulationsergebnisse für die Wartezeit-VFn dieses Beispiels 4.2. Für $i=2$ würde eine Wartezeit $t=48$ s bei optimaler Abfertigung mit einer Wahrscheinlichkeit von 0,9 unterschritten, während die entsprechende Wahrscheinlichkeit der Strategie FCFS nur 0,78 beträgt.
Nun betrachten wir den komplizierteren Fall mit mehr als einem Typ von Anforderungen innerhalb einer Gruppe, wobei einige Typen wenig, die anderen stark streuende Bediendauern haben. Aus dem vorangegangenen Beispiel 4.2 ist bekannt, daß die Abarbeitungsreihenfolge innerhalb einer Gruppe sich nicht auf die charakteristischen Größen (etwa die mittlere Wartezeit und ihre VF) anderer Gruppen auswirkt. Obwohl also auch andere Gruppen (mit anderen unterbrechenden statischen Prioritäten) vorliegen können (vgl. Modell nach Bild 9), dürfen wir jede Gruppe isoliert betrachten [1] und verwenden deshalb zur Betrachtung einer isolierten Gruppe das Modell nach Bild 2:

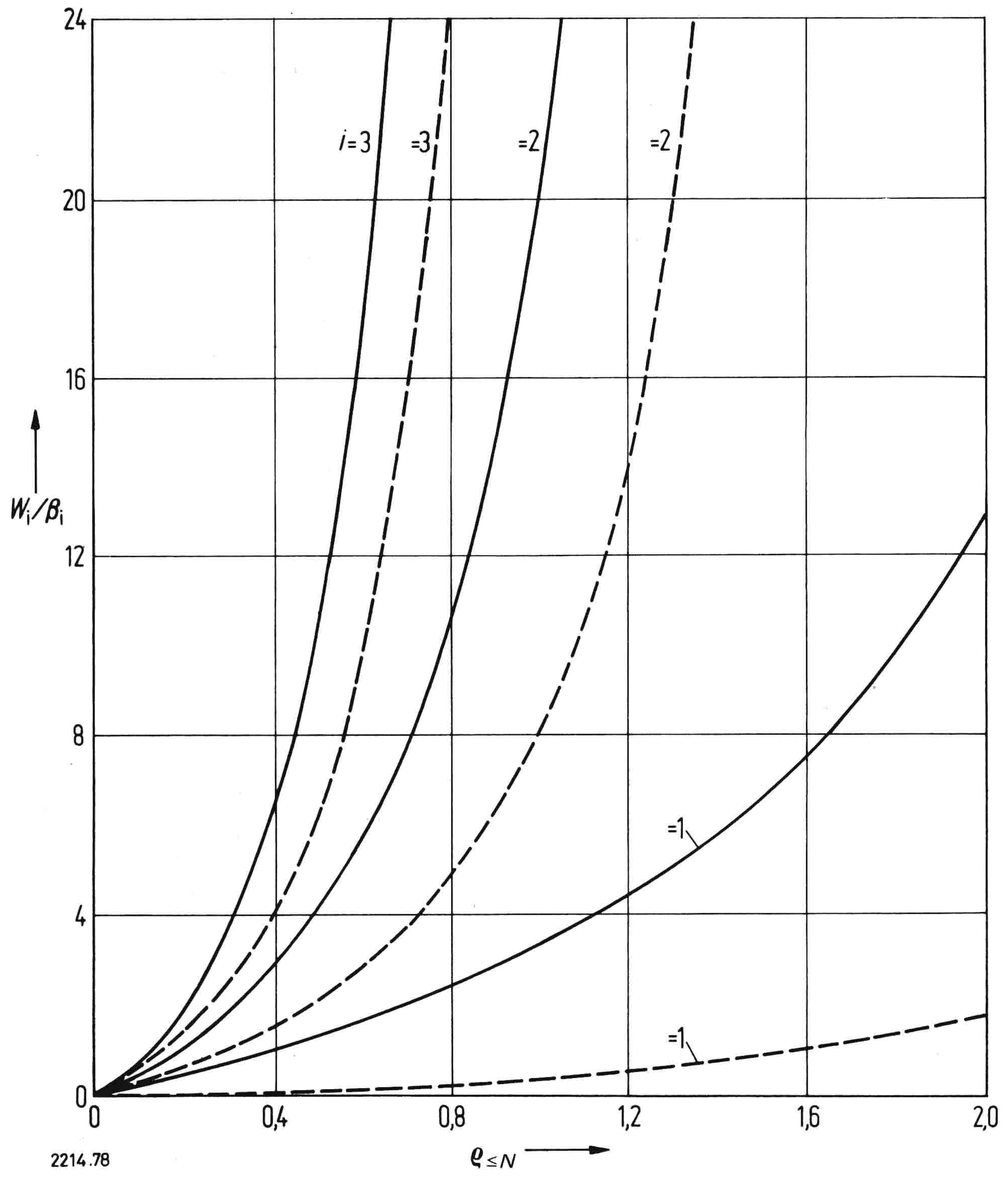


Bild 10: Bezogene mittlere Wartezeit über dem Gesamtangebot. Die durchgezogenen Kurven gelten für FCFS-Abfertigung, die gestrichelten Kurven für optimale Abarbeitung innerhalb der Prioritätsebenen *i*.

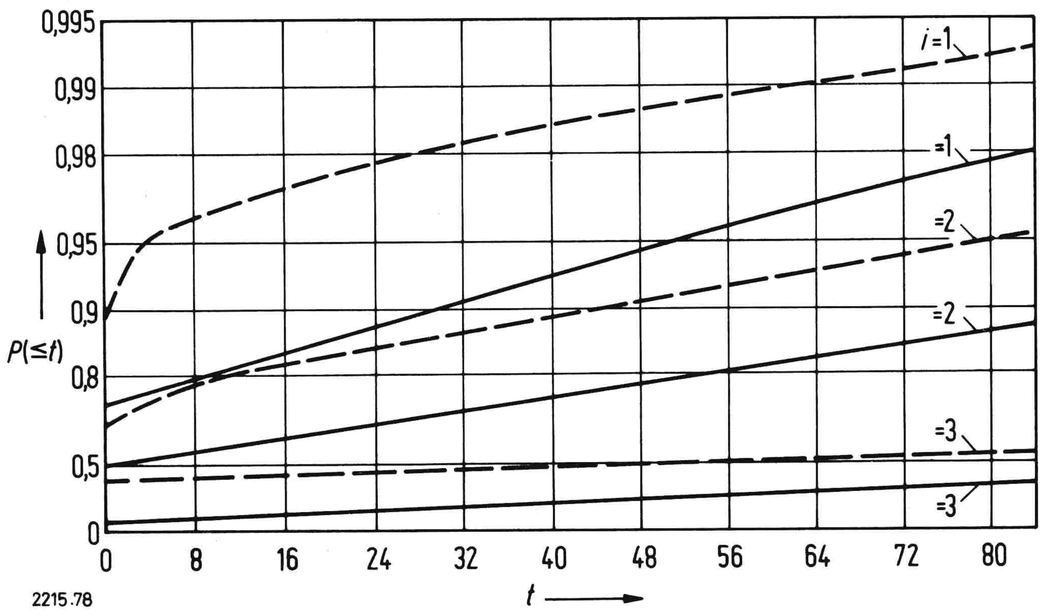


Bild 11: Wartezeit-VFn. Die durchgezogenen Kurven ergeben sich bei FCFS-Bearbeitung in jeder Prioritätsebene, die gestrichelten Kurven bei optimaler Bearbeitung. Gesamtangebot $\rho_{\leq N}=0{,}9$ (vgl. auch Bild 10).

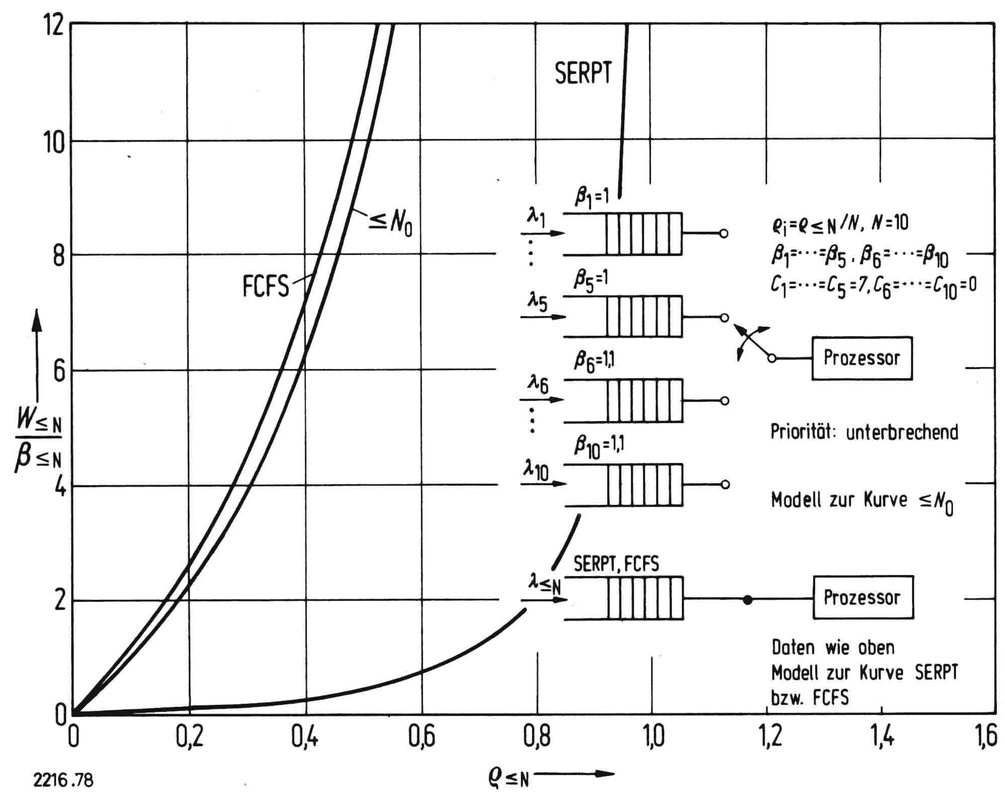


Bild 12: Bezogene gemeinsame mittlere Wartezeit bei „optimalen" unterbrechenden statischen Prioritäten (Kurve $\leq N_0$) und optimalen internen Prioritäten (Kurve SERPT). Zum Vergleich ist das Ergebnis der Strategie FCFS eingetragen.

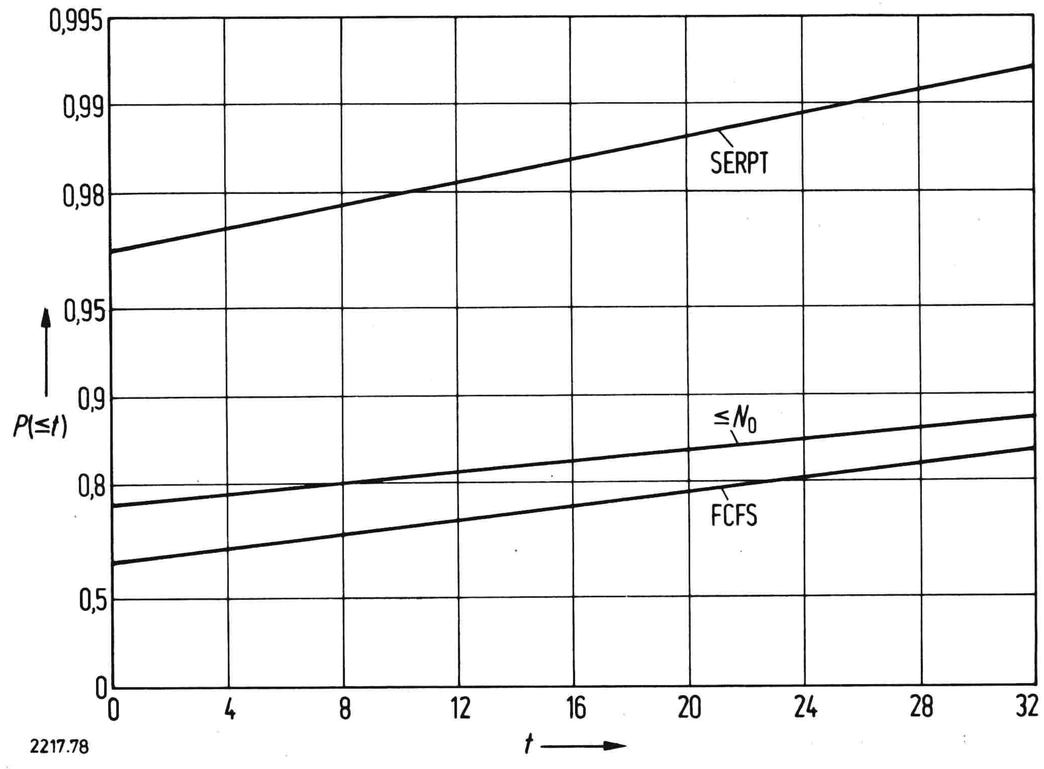


Bild 13: Wartezeit-VFn zum Beispiel von Bild 12. Gesamtangebot $\rho_{\leq N}=0{,}55$.

Beispiel 4.3

$N=10$ Typen von Anforderungen im Modell nach Bild 2, gleiche Angebote $\varrho_i=\varrho_{\leq N}/N$, die Typen 1 bis 5 haben stark streuende gleiche mittlere Bediendauern, $\beta_i=1$, mit einem Variationskoeffizienten $C_i=7$, die Typen 5 bis 10 haben konstante gleiche Bediendauern, $\beta_i=1{,}2$, mit einem Variationskoeffizienten $C_i=0$. Der gemeinsame Bediendauer-Erwartungswert ist $\beta_{\leq N}=1{,}1$.

Verwendet man zur Bearbeitung dieser Gruppe von zehn Typen von Anforderungen statische Prioritäten entsprechend der Regel (1), dann ergibt sich für die gemeinsame mittlere Wartezeit die mit $\leq N_0$ bezeichnete Kurve in Bild 12. Dagegen erhält man bei Anwendung optimaler interner Prioritäten (Strategie SERPT, Zeitscheibenverfahren, vgl. [1]) die mit SERPT bezeichnete Kurve. Bei konstantem Angebot wird dabei die mittlere gemeinsame Wartezeit aller Anforderungen dieser Gruppe um Faktoren kleiner. Auch die Wahrscheinlichkeit für die Unterbietung irgendeiner betrachteten Wartezeit t wird bei internen Prioritäten erheblich größer (Bild 13) als bei statischen Prioritäten. Bemerkenswert ist, daß die Abfertigungsstrategie FCFS am schlechtesten abschneidet.

4.3 Optimierungsziel: Bestmögliche Garantie von Terminen

Wir gehen hier davon aus, daß innerhalb einer Gruppe von Anforderungen ausschließlich Typen von Anforderungen vorliegen, die unterschiedliche Terminforderungen haben. Ziel ist es, die Termine aller Anforderungen innerhalb der Gruppe gleich gut einzuhalten.

Neben diesen Anforderungen können bei einem Realzeitrechner auch noch andere Typen von Anforderungen vorliegen, die zu anderen Gruppen gehören mögen. Die Bearbeitung von konkurrierenden Gruppen wurde bereits besprochen.

4.3.1 Ergebnisse für den ereignisgesteuerten Betrieb

Da Anforderungen hier zu zufälligen Zeitpunkten auftreten, können Termine nur mit gewisser Wahrscheinlichkeit eingehalten werden. Wir betrachten ein Modell entsprechend Bild 2. Zusätzlich zu den dort eingeführten Parametern setzen wir voraus, daß für Anforderungen vom Typ i eine Wichtigkeit v_i fest vorgegeben ist. v_i sei die maximal erwünschte Wartezeit der Anforderungen vom Typ i. Aus der Summe von Ankunftszeitpunkt t_i, Wichtigkeit v_i und tatsächlicher Bediendauer b_i einer Anforderung ergibt sich ihr *erwünschter Fertigstellungstermin*.

Es gibt eine Strategie RU (relative urgency), die unter Beachtung der Wichtigkeiten v_i erreicht, daß die Terminüberschreitungs-Wahrscheinlichkeiten aller Typen von Anforderungen innerhalb einer Gruppe minimal werden. Unterschiedliche Typen von Anforderungen haben dabei gleiche oder verschiedene Ankunftsprozesse und

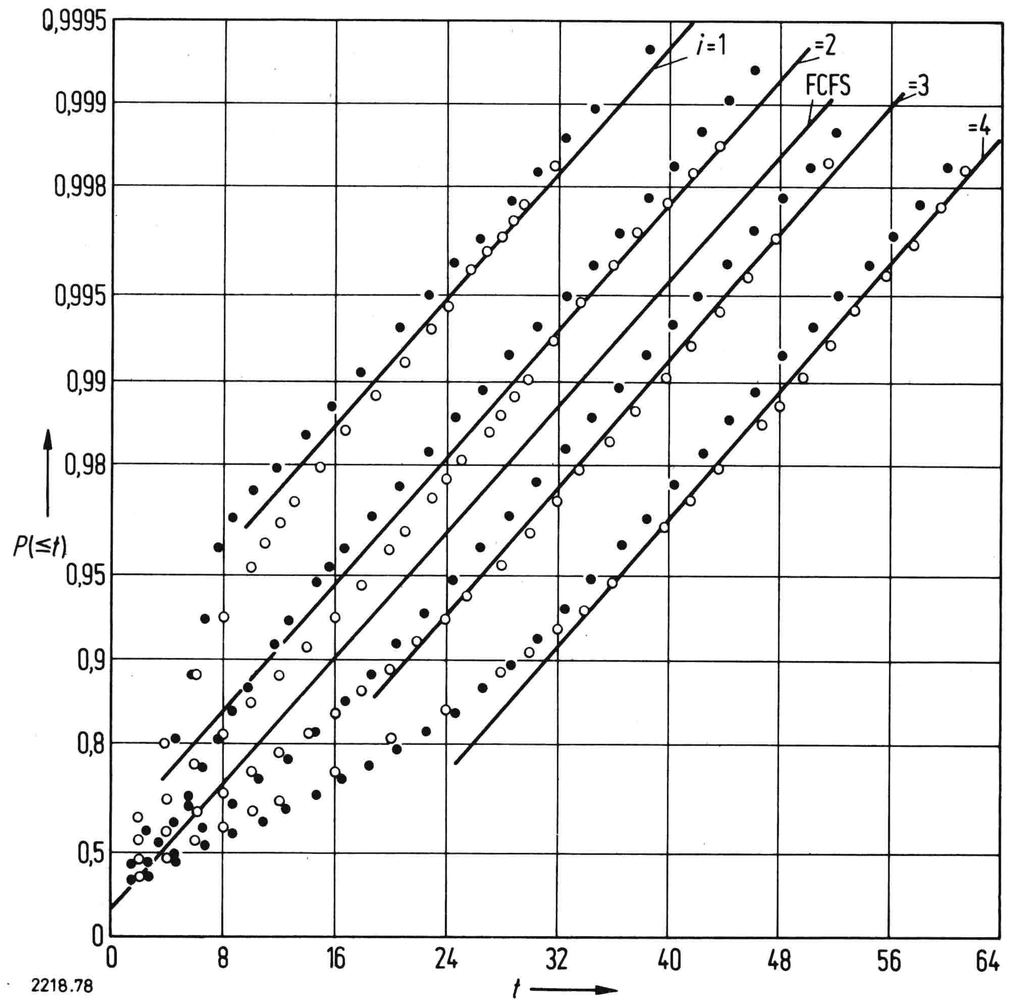


Bild 14: Wartezeit-VFn unter Berücksichtigung von Terminen bei der Strategie RU. Die Punkte gehören zum erstgenannten Bediendauer-Erwartungswert im eingezeichneten Modell, die Kreuze zum zweiten, die durchgezogenen Kurven sind berechnete Näherungen.

Bediendauer-VFn und unterschiedliche Wichtigkeiten v_i. Diese Strategie RU ermittelt die aktuelle Priorität $q_i(t)$ einer Anforderung vom Typ i zum Zeitpunkt t aus der Beziehung $q_i(t)=v_i-w_i(t)$, wobei $w_i(t)$ die dann aufgelaufene tatsächliche Wartezeit der betrachteten Anforderung ist.

Die Ergebnisse für die Termineinhaltungs-Wahrscheinlichkeiten sind bei nicht-unterbrechender Bearbeitung ähnlich gut wie beim Einsatz unterbrechender Prioritäten [1]. Deshalb beschränken wir uns auf den Fall *ohne* Unterbrechungen.

Für das Beispiel 4.1 mit den folgenden beispielhaft gewählten Wichtigkeiten $v_1=15$, $v_2=25$, $v_3=35$, $v_4=45$[3]) erhält man bei der Strategie RU für die $N=4$ betrachteten Typen von Anforderungen Wartezeit-VFn, wie in Bild 14 angegeben. Die durch Punkte (für das Beispiel 4.1 a) bzw. Kreise (für das Beispiel 4.1 b) dargestellten Ergebnisse entstammen einem Simulationsmodell. Die durchgezogenen Linien sind berechenbar. Zum Vergleich ist das Ergebnis der Strategie FCFS eingetragen. Man stellt fest, daß für beide Beispiele 4.1a und 4.1b je die durch v_i gegebenen Termine für alle Typen von Anforderungen mit etwa gleichgroßer Wahrscheinlichkeit $P(w_i>t)=1-P(\leq t)$ überschritten werden. Diese Wahrscheinlichkeit ist etwa gleichgroß für die Beispiele 4.1 a und 4.1 b und etwa gleich $\ddot{U}$, siehe Gl. (2) mit $t_i=v_i$. Man erhält aus Bild 14 für die Beispiele 4.1 a bzw. 4.1 b $\ddot{U}_a=0{,}012$ und $\ddot{U}_b=0{,}017$.

Für dieses Beispiel, aber mit Anwendung statischer Prioritäten, ergeben sich die in Bild 7 (für unterbrechende Prioritäten) und Bild 8 (für nicht-unterbrechende) dargestellten, untereinander ähnlichen Verläufe der Wartezeit-VFn. Der wesentliche Unterschied zu den Ergebnissen der termin-orientierten Strategie RU ist, daß bei letzterer alle Typen von Anforderungen bezüglich ihres Wartezeitwunsches gleich gerecht behandelt werden, während bei statischen Prioritäten einzelne Typen auf Kosten anderer rigoros bevorzugt werden.

Die mittleren Wartezeiten der einzelnen Typen von Anforderungen und ebenso die gemeinsame mittlere Wartezeit $W_{\leq N}$ haben, im Unterschied zu den entsprechenden Ergebnissen bei statischen Prioritäten (Bild 6), alle ihren Pol bei $\varrho_{\leq N}=1$ (vgl. Bild 15). Die Bilder 15a und 15b gelten für dasselbe Belastungsbeispiel 4.1, aber unterschiedliche Paarungen der Wichtigkeiten v_i und Bediendauer-Erwartungswerte β_i. Während mit zunehmendem Index i die β_i im Beispiel 4.1 a zunehmen, nehmen sie im Beispiel 4.1 b ab. Die Anordnung der v_i ist in beiden Beispielen gleich.

[3]) Dieser „Satz" von Wichtigkeitszahlen $\{v_1, v_2, v_3, v_4\}$ entspricht dem Satz $\{v_1+r, v_2+r, v_3+r, v_4+r\}$, man darf also eine reelle Konstante addieren, ohne daß sich die Ergebnisse (z. B. in Bild 14) ändern.

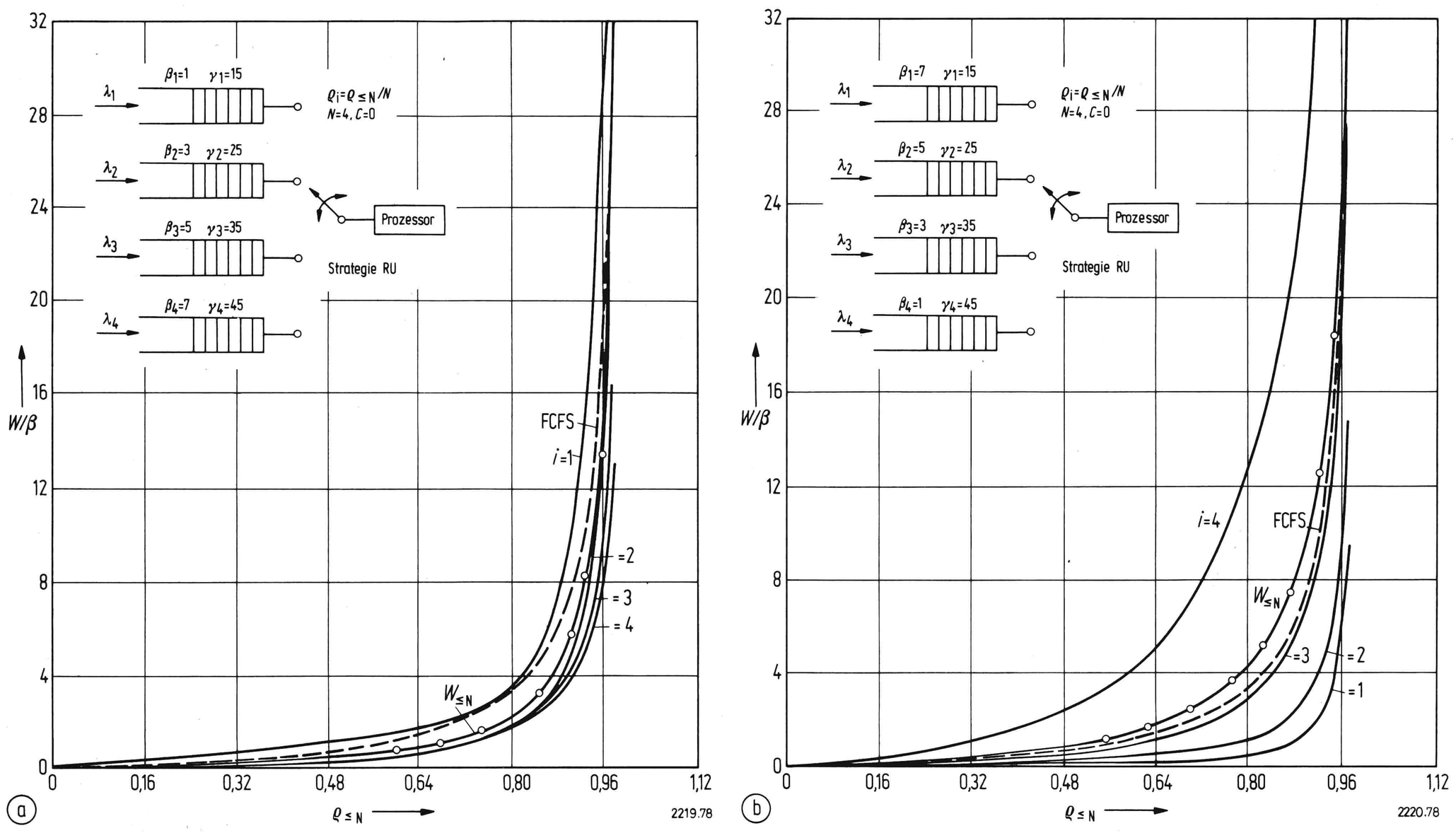


Bild 15: Bezogene mittlere Wartezeit über dem Gesamtangebot bei der RU-Strategie für das Beispiel „4.1 a" (Bild a) und „4.1 b" (Bild b) mit $v_1=15$, $v_2=25$, $v_3=35$, $v_4=45$. Die unterbrochene Kurve gilt für die Strategie FCFS. Die mit ○ markierte Kurve zeigt die gemeinsame mittlere Wartezeit.

Die Kurven FCFS verlaufen identisch gleich. Durch die Vorgabe von Wichtigkeiten verlaufen bei der Strategie RU alle mittleren Wartezeiten in der Tendenz ähnlich wie die Kurve der Strategie FCFS. Das ist bei statischen Prioritäten – Bilder 6a bis 6c – nicht der Fall. Da die Kurven FCFS in den Bildern 6a, b und 15a, b identisch gleich verlaufen, kann man mit ihrer Hilfe den Verlauf der *gemeinsamen* mittleren Wartezeiten $W_{\leq N}$ vergleichen: Die gemeinsame mittlere Wartezeit wächst, wenn man anstelle optimaler, nach Regel (1) zugeteilter statischer Prioritäten Wichtigkeiten v_i einführt und die Strategie RU anwendet, siehe die Bilder 6a und 15a. Sie fällt, wenn man anstelle ungünstigst gewählter statischer Prioritäten (Beispiel 4.1b) Wichtigkeiten einführt und die Strategie RU benutzt, siehe die Bilder 6b und 15b.

Dieses Ergebnis gilt in der Tendenz allgemein, unabhängig vom Belastungsfall. Aus den Resultaten dieses Beispiels kann man keine allgemeingültigen Aussagen über die Optimalität statischer Prioritäten oder der Strategie RU bezüglich einer kleinstmöglichen gemeinsamen mittleren Wartezeit ableiten. Je nach Vergabe der statischen Prioritäten ist ein entsprechendes Modell mit RU-Strategie bezüglich des Minimums von $W_{\leq N}$ über- oder unterlegen, siehe die Bilder 6a und 15a bzw. 6b und 15b.

Die Normierung der mittleren Wartezeit W_i auf den Bediendauer-Erwartungswert β_i hat in Bild 15a zur Folge, daß die Anforderungen vom Typ $i=4$ als besser bedient erscheinen als die vom Typ $i=1$. Das gilt jedoch nur relativ, nämlich im Verhältnis zur mittleren Bediendauer β_i.

Trägt man, anstelle der bezogenen, die mittlere Wartezeit W_i an der Ordinate ab, dann ergibt sich für beide Beispiele 4.1a und 4.1b (mit den genannten v_i) das in Bild 15c gezeigte Ergebnis. Wie bei nicht-unterbrechenden statischen Prioritäten (siehe Bild 6c), sind die mittleren Wartezeiten bei Annahme gleicher Angebote $\varrho_i = \varrho_{\leq N}/N$ unabhängig von der Paarbildung (v_i, β_i). Nur die gemeinsamen mittleren Wartezeiten $W_{\leq N}$, in denen die Häufigkeit des Auftretens von Anforderungen des Typs i berücksichtigt sind, unterscheiden sich, und zwar bei der Strategie RU spürbar weniger als bei statischen nicht-unterbrechenden Prioritäten.

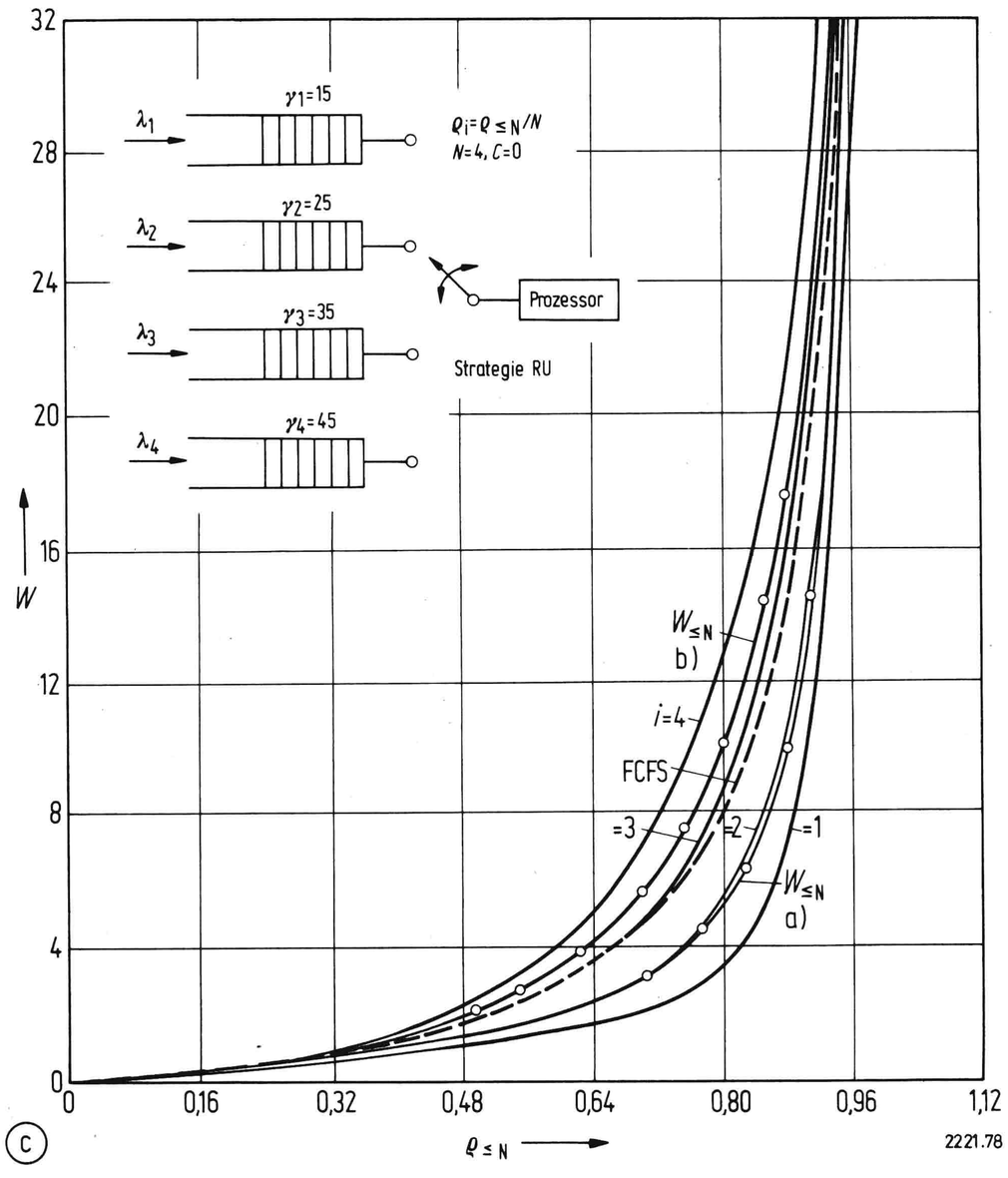


Bild 15c): Mittlere Wartezeit bei nicht-unterbrechender Variante der terminorientierten Strategie RU. Die Ergebnisse für die mittlere Wartezeit W_i (Kurven $i=1$ bis 4) gelten für beide in den Bildern 14a) und c) beschriebenen Belastungsbeispiele. Dagegen sind die zugehörigen gemeinsamen mittleren Wartezeiten $W_{\leq N}$ (Bild 15a) bzw. $W_{\leq N}$ (Bild 15b) verschieden.

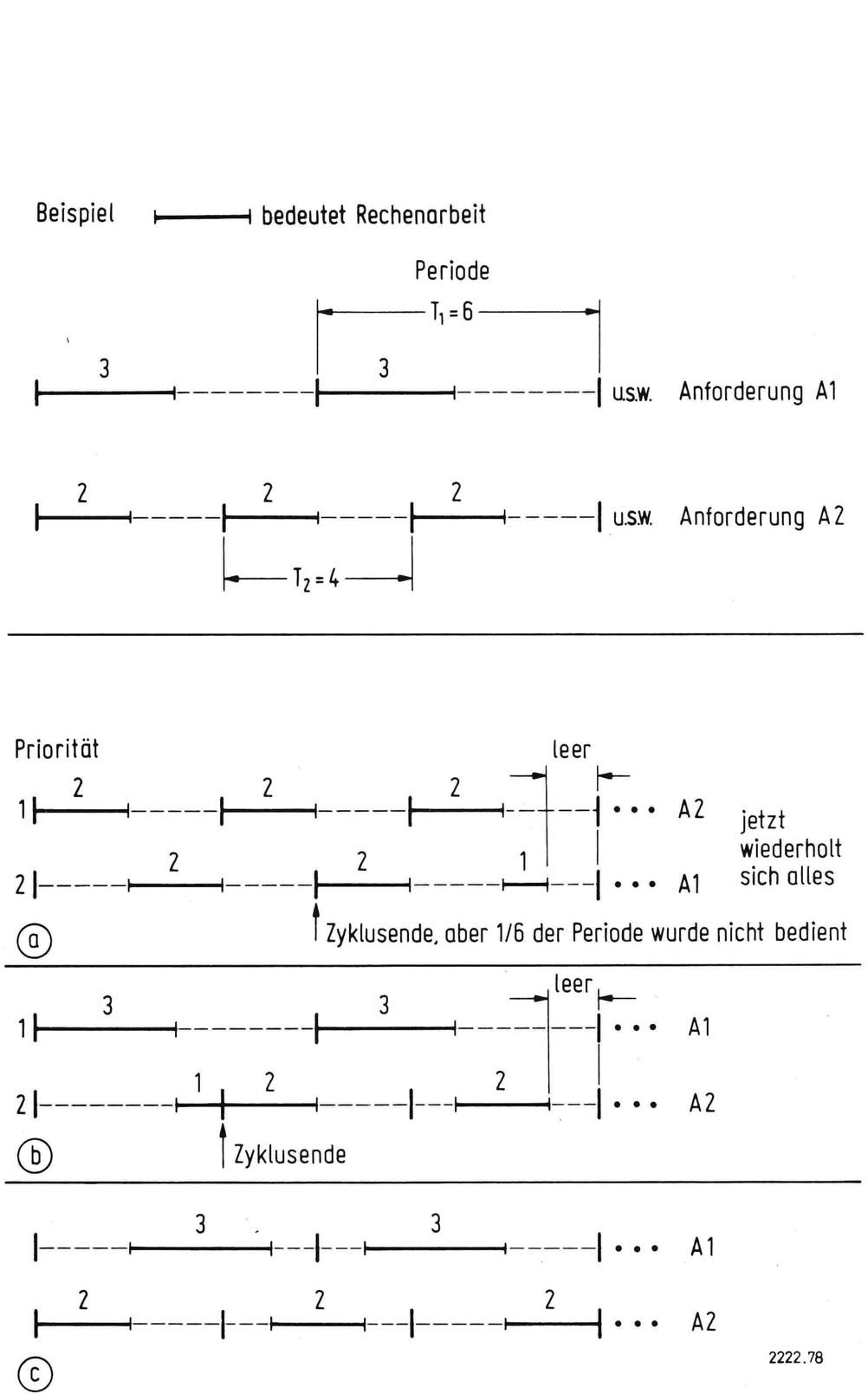


Bild 16: Beispiel „4.4" für die Bearbeitung von zwei periodischen Anforderungen A1 und A2 und kritischer Situation. Periodenanfang und -ende sind durch dicke senkrechte Balken markiert. In den Fällen a) bzw. b) werden optimale bzw. ungünstige statische unterbrechende Prioritäten verwendet, versagen jedoch. Im Fall c) kommt die Strategie RU ohne Unterbrechungen aus und bearbeitet beide Anforderungen fristgerecht.

4.3.2 Ergebnisse für den taktgesteuerten Betrieb

Die oben beschriebene Strategie RU (aber mit unterbrechender Priorität für in Terminüberschreitungs-Gefahr befindliche Anforderungen) hält bei 100%iger Prozessor-Belastung alle Termine ein. Gegenüber der Anwendung optimaler unterbrechender statischer Prioritäten (siehe Abschnitt 4.1.2) wird die Anzahl der Unterbrechungen hier deutlich herabgesetzt, denn nur bei Gefahr einer Terminüberschreitung wird unterbrochen. Für die Realisation ist die (softwaremäßige) Verwaltung aller wartenden Anforderungen nötig. Die augenblickliche (dynamische) Priorität berechnet sich aus „Zeitpunkt der nächsten periodischen Anforderung desselben Typs minus Restbediendauer".

Man kann leicht nachprüfen, ob eine gegebene Kombination periodischer Anforderungen mit Periodendauern T_i und bekannter Bediendauer β_i pro Periode termingerecht bearbeitbar ist. Es muß nach Gl. (4) erfüllt sein

$$\varrho_{\leq N} = \sum_{i=1}^{N} \beta_i / T_i \qquad (5)$$

Verwaltungszeit ist dabei nicht berücksichtigt. Die Arbeitsweise statischer bzw. dynamischer Prioritäten läßt sich leicht vergleichen:

Beispiel 4.4

Modell nach Bild 2, $N=2$, $\beta_1=3$, $\beta_2=2$, $\beta_1/T_1=\beta_2/T_2=0{,}5$, $T_2/T_1=2/3$ (d.h. nicht ganzzahlig teilbare Periodendauern), $\varrho_{\leq N}=1$, siehe Gl. (5).

In Bild 16 oben sind diese Daten graphisch dargestellt. Darunter findet man unter a) optimal nach Gl. (3) zugeteilte statische unterbrechende Prioritäten, unter b) die gegenteilige statische

Prioritätsvergabe und unter c) den Bearbeitungsablauf bei der Strategie RU. In allen drei Fällen ist eine „kritische Situation" unterstellt, nämlich der hin und wieder auftretende besonders unangenehme Fall, daß alle (hier beide) periodischen Anforderungen gleichzeitig ihren Zyklus beginnen.
Im Fall a) unterbricht die Anforderung A2 bei Beginn ihres zweiten Zyklus die Anforderung A1 und verhindert dabei, daß diese noch rechtzeitig bis zu ihrem Zyklusende fertig wird. Im Fall b) wird die Anforderung A1 vorrangig bearbeitet, wobei die Anforderung A2 ihren Termin verpaßt. Typisch für beide Fälle ist, daß im weiteren Verlauf des Bearbeitungsplanes Zeiten auftreten, während denen keine Prozessorarbeit verlangt wird.
Im Fall c) treten keine gegenseitigen Unterbrechungen der Anforderungen auf und dennoch werden alle termingerecht bearbeitet. Streuen die Bediendauern, dann hält die Strategie RU unverändert alle durch die Periodendauern gegebenen Termine am besten ein. Doch kann die Einhaltung aller Termine nur noch mit einer Wahrscheinlichkeit $p<1$ garantiert werden. Diese Wahrscheinlichkeit kann durch Verkleinerung des Gesamtangebotes $\varrho \leq N$ vergrößert werden (und höchstens $p=1$ werden).

5. Optimale Strategien für unkritische Echtzeitanforderungen

5.1 Teilaufgaben, working-set und Konsequenzen für die Wartezeit

5.1.1 Modell mit Hintergrundspeicher

Treten während der Bearbeitung einer Anforderung abwechselnd Transport- und Rechenphasen auf, dann sind die in den Bildern 3 und 4 dargestellten Realzeitrechner-Modelle anwendbar. Wir nennen die *Sequenz aus je einer Transport- und Rechenphase „Teilaufgabe"*. Sieht man von der Möglichkeit ab, einen Realzeitrechner mit einem Massenspeicher mit extrem kurzer Übertragungszeit auszurüsten, dann führt der Transportanteil der einzelnen sequentiellen Teilaufgaben einer Anforderung zu spürbaren Verzögerungen gegenüber der Bearbeitung bei genügend großem Hauptspeicher. Die Wartezeit einer Anforderung besteht jetzt aus Warten auf Zuteilung des Prozessors und Warten auf Ausführung und Beendigung von Transporten, so daß Wartezeiten im Sekundenbereich auftreten. Wir sprechen dann von unkritischen Echtzeitanforderungen.
Die folgenden Aussagen setzen voraus, daß allein dieser Typ von Anforderungen vorliegt. Falls noch andere Gruppen von Anforderungen quasi gleichzeitig bearbeitet werden sollen, treffen die über die Bearbeitung von unterschiedlich wichtigen Gruppen von Anforderungen in Abschnitt 4.1 gemachten Aussagen zu.

5.1.2 working-set

Teilt man einem Rechenprozeß zur Bearbeitung einer Anforderung weniger Zentralspeicherplatz zu als seinem (virtuellen) Adreßraum entspricht, dann entstehen in der Regel Teilaufgaben. Im Zentralspeicher befindet sich dann nur der working-set des Programms. Immer, wenn auf nicht im Zentralspeicher vorhandene Daten zugegriffen werden soll oder ein Puffer keine weiteren Daten aufnehmen kann, wird die Unterbrechung der Prozessorarbeit und die Ausführung eines Hintergrund-Transportes nötig (Rückführung von RK zum HSP in Bild 3). Es ist in diesem Zusammenhang ohne Bedeutung, ob der Transport durch demand-paging- oder demand-segmenting-Techniken mit Unterstützung des Betriebssystems erfolgt, oder ob die Mehrfachverwendung eines begrenzten Zentralspeicherbereiches durch overlay vom Programmierer des betroffenen Prozesses selbst organisiert werden muß. Als äußeres Erscheinungsbild bleibt, daß eine Anforderung in Teilaufgaben zerfällt.

Bild 17 zeigt, wie für einzelne Programme die Zahl $m(x)$ von Hintergrund-Transporten vom in den Zentralspeicher geladenen Anteil x ihres gesamten virtuellen Adressenraumes abhängt. Die Zahl der Teilaufgaben nimmt mit von $x=1$ abnehmender working-set-Größe zunächst nur langsam, später aber sprunghaft zu. Die

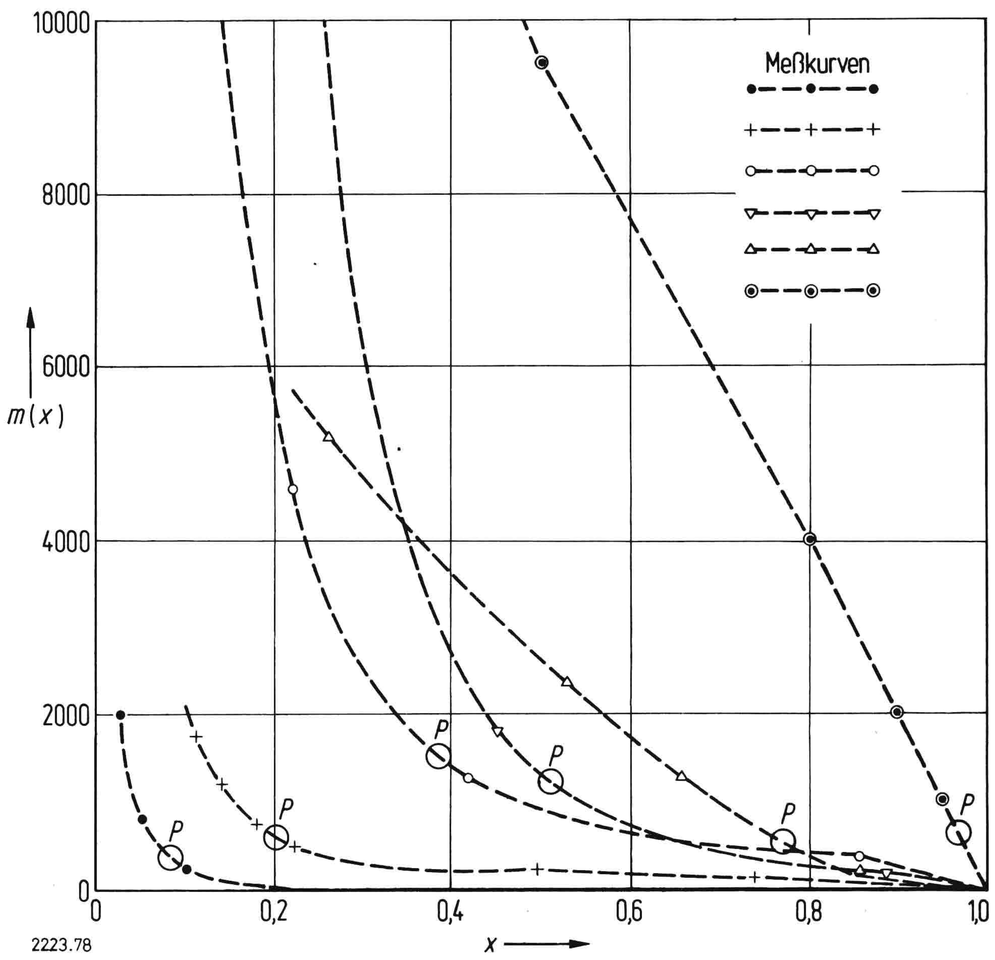


Bild 17: Anzahl der Teilaufgaben $m(x)$ in Abhängigkeit vom in den Hauptspeicher geladenen Anteil x des gesamten virtuellen Adressenraumes für sechs verschiedene Programme. P(=Parachor) bezeichnet den Anteil x, bei dem noch „wenige" Teilaufgaben auftreten und liegt deshalb an der Knickstelle der entsprechenden Kurven.

kritische Größe $x<1$ des working-sets, bei der die Zahl der Zusatztransporte (wegen beschränkten Zentralspeicherplatzes) noch klein bleibt, ist ein individueller Parameter jedes Programmes. Sie läßt sich durch geeignete Programmierung, etwa das Ablegen von zusammengehörigen Befehlen und Daten in gemeinsame Transporteinheiten (Seite, Segment) spürbar verkleinern. Das hier für Anwendungsprogramme (beispielsweise Compiler, FFT, Simulationsprogramm, Lösung von Differentialgleichungen usw.) in Bild 17 dargestellte Verhalten gilt ebenso für Betriebssystem-Programme. Man spricht von gutem Lokalitätsverhalten eines Programmes, wenn der working-set so groß ist, daß man sich rechts von der Knickstelle P seiner Kurve nach Bild 17 befindet (P=Parachor).

Das Zerfallen einer Anforderung in Teilaufgaben bei Bearbeitung in einem zu kleinen Zentralspeicherbereich beobachtet man sowohl im Einprogramm- als auch im Mehrprogrammbetrieb. Man kann zeigen, daß die Zerstückelung einer Anforderung in Teilaufgaben unter dem Gesichtspunkt der Auslastung des Rechnerkerns kein Nachteil sein muß: Im Mehrprogrammbetrieb kann unter Umständen ein Programm den Rechnerkern nutzen, während ein anderes Programm gerade transportiert.

5.1.3 Working-set-Größe und Wartezeiten

Zwei wichtige Dinge sollten beachtet werden:

I. Wenn die aus einer zu klein gewählten working-set-Größe resultierende Transportarbeit nicht vollständig durch Rechenarbeit für andere Anforderungen überlappt werden kann, dann steht der Rechnerkern zeitweise leer, unabhängig von der Zahl der im Mehrprogrammbetrieb bearbeiteten Programme. Leerzeit des Rechnerkerns wirkt sich als Wartezeit aller in Arbeit befindlichen Programme aus.
Eine vollständige Überlappung ist wegen der meist statistisch schwankenden Dauern von Rechen- und Transportphasen theoretisch nur bei unendlich vielen quasi gleichzeitig bearbeiteten Programmen möglich. Zusätzlich muß die Bedingung erfüllt sein, daß die mittlere Transportzeit von Teilaufgaben kleinergleich ihrer mittleren Rechnerkern-Belegungsdauer ist. Bei den heutigen für Realzeitrechner verwendeten Platten-Hintergrundspeichern dauert ein Transport etwa 20 bis 70 ms. Die Rechenzeit einer Teilaufgabe muß also im Mittel ebenso groß sein. Ist das erfüllt, dann erreicht man eine gute Überlappung

der Arbeit von Kanal und Rechnerkern schon mit wenigen Programmen gleichzeitig, indem man optimale Abfertigungsstrategien verwendet. Bei (fast) vollständiger Überlappung von Transport durch Rechenarbeit wird die mittlere Wartezeit (fast) so klein wie im Modell ohne Transporte, zuzüglich der eigenen Transportzeit im Modell mit Transporten.

II. Ein hoher Multiprogramminggrad, der oft durch starke Platzbeschränkungen für die einzelnen Programme im Zentralspeicher erkauft wird, lohnt sich bezüglich der Wartezeitminimierung nur, wenn die in I.) erwähnte Überlappung gewährleistet ist *und* eine optimale Rechnerkern-Zuteilungsstrategie angewandt wird, die aus der Auswahlmöglichkeit gleichzeitig auf den Rechnerkern wartender Teilaufgaben Nutzen zieht. Verzichtet man innerhalb einer Gruppe von gleich wichtigen Anforderungen auf diese Auswahlmöglichkeit und wendet, wie oft innerhalb einer Prioritätsebene antreffbar, die Strategie FCFS an, dann bringt der Mehrprogrammbetrieb gegenüber dem Einprogrammbetrieb meistens nur Nachteile: die mittlere Wartezeit bleibt nicht unverändert, sondern wächst. Man halte sich vor Augen, daß ein hoher Prozentsatz aller Transporte überhaupt nur durch die beschriebene, für den Mehrprogrammbetrieb als Voraussetzung notwendige Platzbeschränkung für die einzelnen Programme zustande kommt.

Mehrprogrammbetrieb mit Anforderungen verschiedener statischer Priorität führt in der Regel gegenüber Einprogrammbetrieb mit jeweils der wichtigsten Anforderung zu unerwünschter Nivellierung: die Wartezeiten unterschiedlich wichtiger Anforderungen rücken dabei zusammen.

Nutzt man die oben angesprochene Auswahlmöglichkeit eines aus mehreren auf den Rechnerkern wartenden Programmen, dann kann sich Mehrprogrammbetrieb zur Verkleinerung der mittleren Wartezeit einer Gruppe gleich wichtiger Anforderungen spürbar lohnen – und zwar nur dann.

5.1.4 Hintergrund-Transporte wegen zu kleinem working-set sind Wartezeiten

Aus dem Gesagten folgt, daß Hintergrund-Transporte möglichst vermieden werden sollten. Es gibt nur unter glücklichen oder durch eine überlegte Strategie herbeigeführten Umständen gerade immer *eine* Anforderung, die den Rechnerkern während eines Transportes belegt. Transportzeiten sind in vielen Fällen fast vollständig Beiträge zu Wartezeiten der eigenen und zugleich aller anderen wartenden Anforderungen. Diese Aussage gilt für Grund- wie Anwendersoftware. Nach jedem Transport sollte eine Rechenphase folgen, die im Mittel mindestens ebenso lange dauert. Ist das für manche Anforderungen nicht erfüllt, dann müssen gleichzeitig sehr rechenintensive *gleichwichtige* Anforderungen vorliegen, bei denen die Dauer einer Rechenphase ein Vielfaches der Dauer einer Transportphase ist. Sind beide Bedingungen nicht erfüllt, dann sollte den einzelnen Anforderungen mehr Hauptspeicherplatz zugewiesen werden, um die Transporthäufigkeit zu verkleinern und die Dauer von Rechenphasen zu vergrößern. Das führt im Grenzfall, wenn man die absolute Zentralspeichergröße unverändert läßt und nur den Multiprogramming-Grad senkt, zum Einprogrammbetrieb und zahlt sich durch eine Abnahme der Wartezeiten aus.

Maßnahmen zur Verkleinerung der Transporthäufigkeit sind u. a. bei Grund- und Standardanwendungssoftware:

* Residenz vergrößern,
* Messungen zur Feststellung, welche residenten Teile Ballast darstellen, und Entfernung dieses Ballastes,
* Vergrößerung von Puffern.

Messungen sind auch schon deshalb ratsam, um die Transporthäufigkeit einzelner Prozesse zu ermitteln. Erfahrungsgemäß verschätzt man sich bei der diesbezüglichen Beurteilung selbstentwickelter Programme leicht um Faktoren.

Die Transportdauer ist (unter Berücksichtigung der meist recht großen mittleren Zugriffszeit der Platten-Hintergrundspeicher) praktisch vorgegeben. Es muß abgeraten werden, zu kleine Datenblöcke zu übertragen, da sonst der Nutzanteil eines Transports gegenüber der Gesamtdauer verschwindet. Den Transport von Datenblöcken in Vielfachen von K-Worten ist ratsam. Kleinere Blöcke sind im Einzelfall dann sinnvoll, wenn größere Blöcke sinnlose Information darstellen würden.

Anschließend werden Beispiele behandelt und die beschriebenen Zusammenhänge anhand von Modellen demonstriert.

5.2 Optimierungsziel: Wartezeitgünstige Abfertigung einer Gruppe von Anforderungen

Das wichtigste Optimierungsziel für Anforderungen, bei deren Bearbeitung Transporte auftreten, ist die mittlere Wartezeit. Die Aussagen von Abschnitt 4.2 treffen hier zu; zusätzlich sind jetzt Transporte zu berücksichtigen.

Wir betrachten das Modell nach Bild 3. Es ist nur zur Behandlung *einer* Gruppe von gleichwichtigen Anforderungen geeignet. Anforderungen haben die Ankunfts- (und Fertigstellungs-)-rate λ und die mittlere Bediendauer β. Das Produkt $\lambda\beta=\varrho$ heißt, wie bisher, Angebot und ist die Rechnerkernauslastung. Die Bearbeitung jeder Anforderung besteht aus der sequentiellen Ausführung von im Mittel z Transport- und Rechenphasen mit den mittleren Dauern t und r. Jede Anforderung besteht aus einer oder mehreren Teilaufgaben. Das Verhältnis von mittlerer Transportzeit zu mittlerer Rechenzeit einer Teilaufgabe sei $h=t/r$. Der Hintergrundspeicher des Modells sei unbegrenzt groß, der Zentralspeicher habe die konstante Zahl n von Programmplätzen für Anforderungen. Also ist n der Multiprogramming-Grad. Das Modell hat nur einen Transportkanal zwischen Zentral- und Hintergrundspeicher.

Die Annahme einer konstanten Zahl n ist eine grobe Vereinfachung der Wirklichkeit, reicht aber zur Demonstration des Einflusses von n auf die mittlere Wartezeit aus. Auf jedem Programmplatz hat ein working-set seines Prozesses Platz. Je kleiner der im working-set zugreifbare Adressenausschnitt eines Prozesses ist, um so größer wird in der Regel die Zahl von Teilaufgaben pro Anforderung. Dabei nimmt das Transportzeit/Rechenzeit-Verhältnis $h=t/r$ zu, denn die gesamte Bediendauer β bleibt unverändert, wird aber in zunehmend mehr Rechenphasen der Dauer r zerstückelt. Die Transportzeit pro Teilaufgabe kann wegen der großen mittleren Zugriffszeit rotierender Hintergrundspeicher als näherungsweise konstant angesehen werden.

Kann der Rechnerkern verhältnismäßig lange mit den im working-set vorhandenen Daten arbeiten, dann spricht man von gutem Lokalitätsverhalten. In der Praxis ändert sich das Lokalitätsverhalten eines Prozesses während seiner Ausführung und ist für verschiedene Prozesse verschieden. Bezieht man diese Feinheiten in die Modellierung ein, dann kommt man zu in der Tendenz gleichen Ergebnissen wie wenn man nur ein einfacheres Modell mit Programmplätzen konstanter Größe unterstellt hätte.

Bild 18 zeigt die Abhängigkeit des Angebotes ϱ vom Transportzeit/Rechenzeit-Verhältnis h, wie sie sich bei der Unterteilung eines Zentralspeichers in n Programmplätze ergibt. Die erreichte mittlere Wartezeit W ist konstant. Geht man etwa von $n=2$ Plätzen und einem Angebot $\varrho=0{,}8$ aus und verkleinert den Zentralspeicher (was möglicherweise einen Anstieg von h um den Faktor 2 zur Folge haben kann) dann muß man das Angebot auf $\varrho=0{,}55$ senken, um eine unverändert mittlere Wartezeit W zu erhalten. Oder vergrößert man n von 2 auf 3 und läßt das Angebot $\varrho=0{,}8$ und die Hauptspeichergröße unverändert, dann muß jeder working-set auf 2/3 der ursprünglichen Größe schrumpfen. Dabei wird das Verhältnis h sicher wachsen. Nur wenn $h=0{,}75$ resultiert, darf das Angebot $\varrho=0{,}8$ bleiben, und W bleibt unverändert gleich. Andernfalls muß ϱ geändert werden oder W ändert sich. Die in Bild 18 gezeigten Ergebnisse gelten bei Voraussetzung einer optimalen Rechnerkern-Zuteilungsstrategie, während der Kanal die Strategie FCFS anwendet. Prinzipiell liegen die Verhältnisse aber unabhängig von den angewandten Strategien ähnlich.

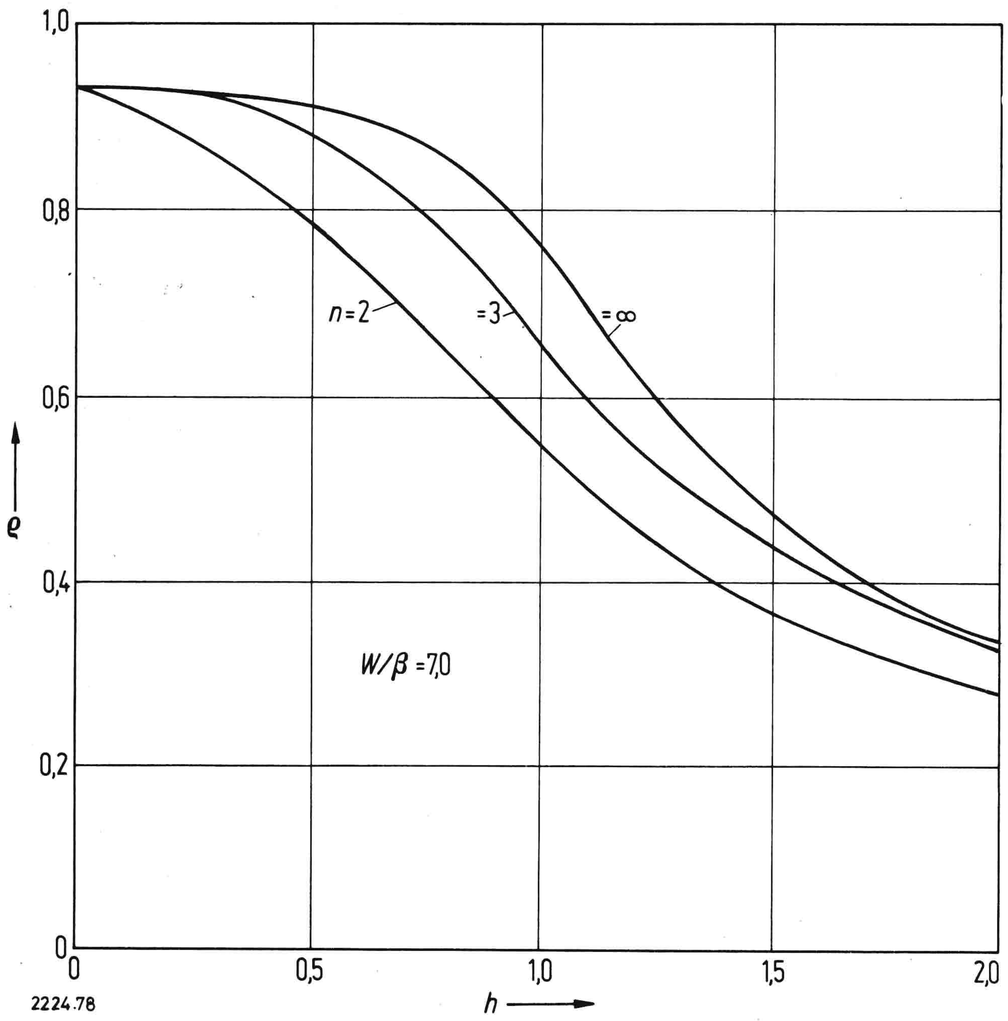


Bild 18: Zulässiges Angebot ρ in Abhängigkeit vom Transportzeit/Rechenzeit-Verhältnis h bei konstanter mittlerer Wartezeit W. Parameter ist die Zahl der Programmplätze im Modell.

Man sollte folgende *Faustregel* beachten:
ist $n \leqq 3$, dann sollte $h \leqq 0{,}6$ sein
ist $n > 3$, dann sollte $h \leqq 1$ sein.
Ergibt sich aufgrund des gewählten Wertes für n ein Verhältnis $h > 1$, dann sollte n verkleinert werden.
Die im Abschnitt 4 behandelten Modelle mit *un*begrenztem Zentralspeicher und *ohne* Kanal sind hier enthalten, wenn man das Transportzeit/Rechenzeit-Verhältnis $h = 0$ setzt. Dann spielt die Zahl n der Plätze keine Rolle. Geht man also von einem Modell mit $h = 0$ auf ein Modell mit $h > 0$ über, dann muß ϱ verkleinert werden, um die bei $h = 0$ erzielte mittlere Wartezeit W aufrecht zu erhalten. Ist die Verkleinerung von ϱ nicht möglich, dann wächst mit zunehmendem h die mittlere Wartezeit W stark an (siehe Bild 19). Soll beispielsweise $W/\beta = 4$ bleiben, während h von 0 auf 0,6 wächst, dann muß ϱ von 0,85 auf 0,63 sinken.

Wäre in Bild 19 n nicht 2, sondern 3 oder größer, dann würde sich die Bezifferung der Kurven mit $h > 0$ ändern: die Werte für h wären bei unverändertem Kurvenverlauf etwas größer. Einen

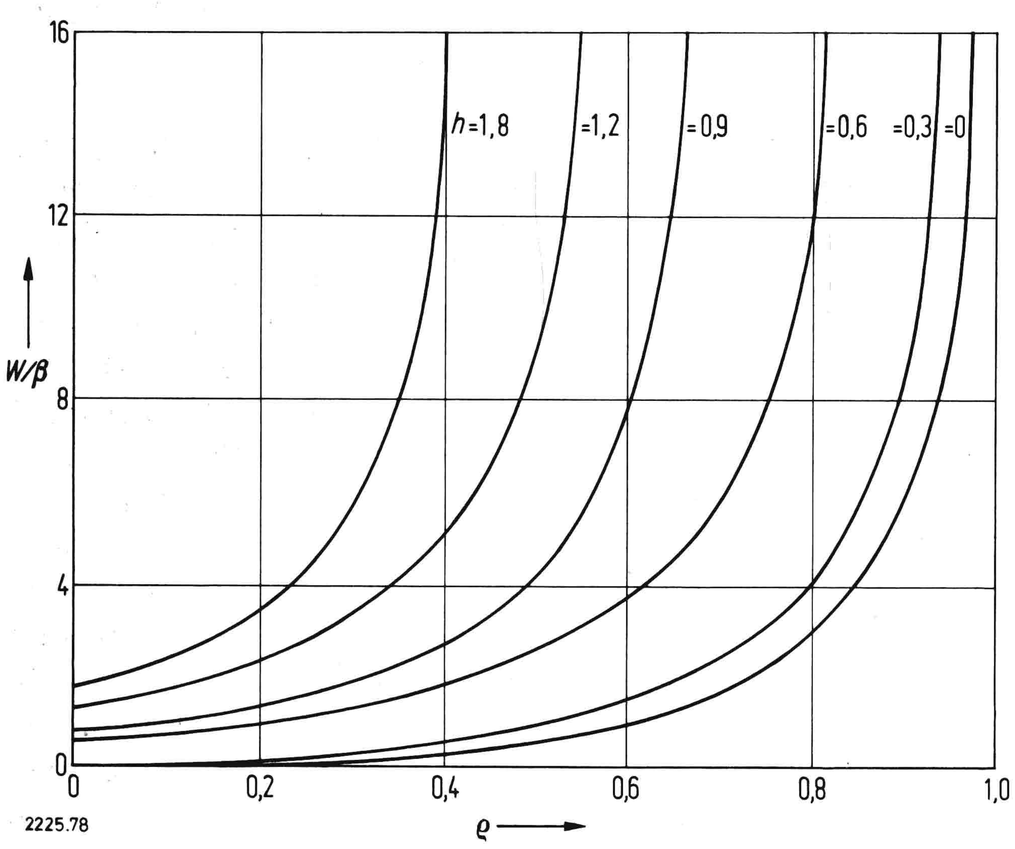


Bild 19: Bezogene mittlere Wartezeit über dem Angebot für ein Modell mit $n = 2$ Plätzen. Parameter ist das Transportzeit/Rechenzeit-Verhältnis h. Modell wie Bild 18.

deutlich größeren Einfluß kann die Abfertigungsstrategie des Rechnerkerns haben: hier gilt prinzipiell dasselbe wie in Bild 12 dargestellt. Dagegen spielt die Strategie am Kanal für Hintergrundtransporte in der Regel keine große Rolle, denn die Transportdauern streuen meist nur wenig.

5.3 Optimierungsziel: Uneingeschränkte Bevorzugung von Anforderungen des Typs i gegenüber anderen vom Typ j

Auch für unkritische Echtzeitanforderungen gelten prinzipiell die in Abschnitt 4.1 gemachten Aussagen weiter. Transporte vergrößern die Ergebnisse für die charakteristischen Größen. Sonst ändert sich nichts Wesentliches.

Das Einbeziehen von Transporten erschwert die Berechnung entsprechender Modelle oder macht sie unmöglich, hat aber auf die günstigsten Strategien für die Zuteilung des Rechnerkerns kaum Einfluß. Es ist offensichtlich, daß Transporte wichtigerer Anforderungen uneingeschränkten Vorrang vor unwichtigeren haben sollten. Das ist im Modell nach Bild 4 berücksichtigt. Für solche Modelle erhält man Ergebnisse wie in den Bildern 5 bis 8 und 10 dargestellt.

5.4 Optimierungsziel: Bestmögliche Garantie von Terminen

Geht man davon aus, daß nicht nur Rechnerkernphasen, sondern auch Transportphasen entsprechend der aus den Terminforderungen der Anforderungen (über ihre Wichtigkeiten v_i) berechneten dynamischen Prioritäten abgefertigt werden, dann sind den in Abschnitt 4.3 besprochenen, für kritische Echtzeitanforderungen erhaltenen Ergebnissen vergleichbare Resultate zu erwarten. Die Beachtung von Terminwünschen ist also auch für unkritische Echtzeitanforderungen möglich und eine optimale Strategie bekannt.

6. Vergleich von statischen, dynamischen und internen Prioritäten

Entscheidende Vorteile statischer Prioritäten sind ihre bestechende Einfachheit und der daraus resultierende geringe Entscheidungsaufwand. Außerdem wird die Ablaufsteuerung mit Hilfe statischer Prioritäten von der Hardware sehr vieler Realzeitrechner unterstützt. Dynamische und interne Prioritäten müssen dagegen softwaremäßig realisiert werden, und dabei läßt sich ein spürbarer Zeitaufwand je Zuteilungsentscheidung nicht unterschreiten. Er entsteht durch die Ermittlung der aktuellen Priorität der wartenden Anforderungen und die Anwendung einer Auswahlstrategie. Um diesen Verwaltungsaufwand gering zu halten, müssen *solche* Strategien mit dynamischen oder internen Prioritäten verwendet werden, bei denen Entscheidungen *selten* notwendig sind.

Neben diesem Verwaltungsaufwand gibt es bei allen Arten von Prioritäten den Unterbrechungsaufwand, der beim Unterbrechen der Bearbeitung einer Anforderung durch eine wichtigere entsteht. Eine Unterbrechung kann bei dynamischen und internen Prioritäten die Folge einer Zuteilungsentscheidung sein.
Für terminorientierte (dynamische) Prioritäten ergibt sich aus Abschnitt 4.3, daß Unterbrechungen überflüssig bzw. in der Version der Strategie RU mit Unterbrechungen [1] sehr selten werden. Gegenüber statischen unterbrechenden Prioritäten wird also Unterbrechungsaufwand gespart. Die Zuteilungsentscheidung erfordert bei der terminorientierten nicht-unterbrechenden Strategie RU je Anforderung ihr termingerechtes Einketten in die Warteschlange wartender Anforderungen. Dieses Einketten wird zweckmäßig nach Fertigstellung der gerade in Arbeit befindlichen Anforderung für alle währenddessen eingetroffenen Anforderungen erledigt und spart so Unterbrechungen ein. Bei hoher Ankunftsrate λ, also kleiner mittlerer Bediendauer β je Anforderung kann dieser Verwaltungsaufwand dennoch spürbar sein [3].
Interne Prioritäten sind gegenüber statischen Prioritäten nur vorteilhaft, wenn die Bediendauern der Anforderungen stark streuen

(stärker als eine negativ exponentielle Verteilung). Der Unterbrechungsaufwand interner Prioritäten ist in der Regel höher als bei unterbrechenden statischen Prioritäten; außerdem tritt noch Verwaltungsaufwand zur Vorbereitung und Durchführung von Zuteilungsentscheidungen auf. Man kennt auch hier praktisch optimale Strategien, die wenig unterbrechen und geringen Verwaltungsaufwand benötigen [1]. In manchen Belastungsfällen ist die Verbesserung der gegebenen Optimierungsziele durch interne Prioritäten gegenüber statischen Prioritäten so erheblich, daß man diesen Verwaltungsaufwand leicht erträgt. In anderen Fällen ist eine sorgfältige Analyse und Einbeziehung zu erwartender Verwaltungs- und Umschaltzeiten notwendig.

NE 33 – Ulm
Eingang 15. 9. 78

Schrifttum:

[1] B. Walke: Realzeitrechner-Modelle, Theorie und Anwendung. Reihe Datenverarbeitung, Oldenbourg-Verlag München 1978

[2] B. Walke: A multi-entrance priority queue with waiting-time optimal servicing. Submitted to 8th Int. Teletraffic Congress, Torremolinos, Spanien, 1979. Erhältlich über den Autor

[3] B. Walke: Prozeßrechner: Statische und terminorientierte Prioritäten im Vergleich. Elektron. Rechenanl. 20 (1978) H. 6, S. 283–289

[4] B. Walke: Strategien zur Leistungssteigerung von Rechenanlagen. Angewandte Informatik 6/76, S. 235–242 (Vieweg-Verlag Braunschweig)

[5] W. Rosenbohm, B. Walke: ‚Waiting-time distributions for deadline-oriented servicing' Proceed. 4th Int. Sympos. Modell. Performance Eval. Comp. Syst; Feb. 1979, Wien-Laxenburg, North-Holland Publishing Company, Amsterdam, New York 1979

[6] E. G. Coffman, P. J. Denning: Operating Systems Theory. Prentice Hall Series in Automatic Computation; Prentice Hall, Inc., Englewood Cliffs, New Jersey 1973