PROCESSOR PRIORITY MODEL WITH DIFFERENT USER TASKS AND OPERATING SYSTEM PHASES

Wolfgang Kraemer

Institute of Switching and Data Technics
University of Stuttgart, Germany

ABSTRACT

A processor model is considered with different types of user tasks as well as different operating system phases. The processing of user code (in problem state) and the processing of operating system code (in supervisor state) are scheduled by interrupt schemes and by dispatching rules. The operating system phases may consist of interrupt handling, scheduling, channel program construction, paging overhead functions... There are m different types of user tasks with different dispatching priorities. The operating system phases initiated by the user tasks are running with higher priority than the associated user tasks. This results in a single server queueing model with feedbacks into higher (and lower) priority classes. Such a model is analyzed under the assumptions of

- a Poisson input process of arriving I/O-interrupt requests and
- general distribution functions for the processing times of user and system tasks.

Main results are the waiting times in the different queues of the operating system as well as response times for the different user tasks.
Numerical results are shown and compared with those obtained by simpler models.

1 INTRODUCTION

Queueing models for multiprogrammed computer systems have achieved great importance for the determination of global performance values of systems with several programs in main storage. For a fixed degree

Presented at the International Symposium on Computer Performance,
Modeling, Measurement, and Evaluation, Yorktown Heights,
New York 1977.
Proceedings published by: North-Holland, Publishing Company,
Amsterdam, New York, Oxford, S. 305-325.

of multiprogramming these models are so-called closed or cyclic models, but also so-called open models are considered for a variable degree of multiprogramming. They are used to calculate the different utilizations of the servers (I/O devices, processor(s)) as well as the associated queue lengths or waiting times.

The central part of such queueing networks is represented by a central processing unit CPU ("central server model", /4/). The utilization of the CPU is a most important measure for system throughput, which - often together with response time constraints - normally is tried to be maximized. This can be done e.g. by different scheduling disciplines for the CPU, denoted in the following as "processor".

The most simple discipline is FIFO (first-in, first-out) between all user tasks, which leads to a standard single server FIFO model for the processor.

In case of time-slicing, it is often more suited to apply processor models with the so-called processor sharing discipline (see also /2/). This discipline automatically favours jobs with short CPU times at the cost of longer ones. Fortunately, this processor discipline is included in queueing networks with "local balance" or "product solution" /3,15/, for which a universal queueing analysis software tool QNET 4 has been implemented /16/.

To increase system throughput, very often fixed priorities are assigned to different user tasks, i.e. the different user tasks reside e.g. in different partitions with different priorities. Normally highest throughput is achieved when high CPU dispatching priority is given to jobs or user tasks with high I/O activity.

To calculate the times these different jobs are in the system, it is necessary to determine the different flow or response times of the central processing unit. For this purpose, ordinary priority models (e.g. preemptive resume) cannot be used generally, since for the scheduling of the operating system queues the priority of the associated user task often is not considered. This, naturally, is useful since operating system phases are used by all user programs concurrently.

The consideration of system overhead can be done e.g.

- by defining global processor occupations including overhead (e.g. /13/), thus preserving a single queue model or
- by explicit consideration of operating system phases with associated queues and priorities, thus leading to single server queueing models with multiple queues, priorities and feedbacks.

The subject of this paper is a separate processor model with several operating system phases of different priorities and with several types of user tasks having different dispatching priorities.

The aim of the queueing analysis of this model is

- to determine the influence of system overheads on the response times of user tasks
- to calculate waiting times within internal operating system queues
- to investigate the influence of the grade of interruptibility of operating system phases (i.e. of the proportion of supervisor code running enabled)
- to compare the results for the response times of the different jobs with response time results of simpler models

It is possible to consider and use this model as a CPU module within an (open) network of a total computer system, also representing the I/O subsystem (disks, drums, channels...), since I/O scheduling often is done without regarding user priorities.

2 PROCESSOR MODEL

2.1 Definition of User Task Phases

The total flow of a batch job or also transaction through a multiprogrammed computer system can be described essentially by a sequence of alternating occupations of the central processor and the I/O devices. Each time a program has to perform an access to an I/O device, a switch into the operating system is necessary to construct e.g. appropriate channel programs and to perform queue scheduling for the I/O system as well as to reschedule the processor itself. Such I/O accesses may be accesses to user files in case of user specified I/O (READ/WRITE) or may be accesses to the page data set in case of a page fault in virtual storage systems.

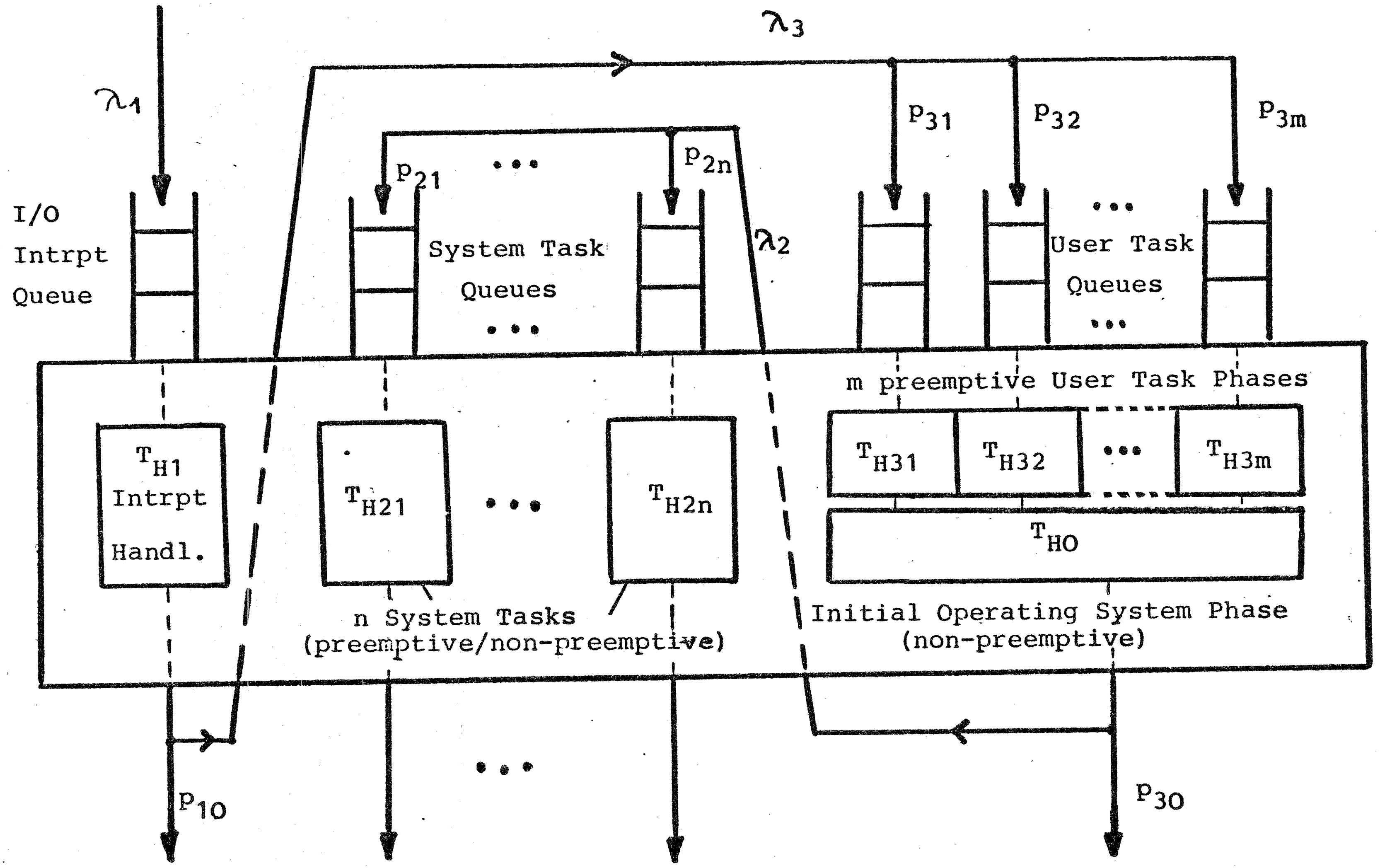

Fig.1: Processor model with operating system processing (levels 0,1,2) and user task processing (level 3). Levels 2 and 3 with sublevels. (λ_i=arrival rates, T_{Hi}=service times, p_{ij}=branching probabilities)

The phases of a user task between such two subsequent switches into supervisor state for performing I/O here are called user task phases. The mean pathlength of such a user task phase can be obtained by dividing the total number of user instructions performed during processing a job in problem state by the total number of its physical I/O accesses.

2.2 Structure of the Processor Model

Figure 1 shows the structure of the processor model with its different queues, priority levels and feedbacks.
The service boxes within the processor represent the processing of user or operating system (supervisor) code. They are labeled for a level i with their service or holding times T_{Hi}, resulting from associated pathlengths and processor speeds (MIPS values).
The model contains queues at different levels

- the I/O-interrupt queue at level 1
- n system task queues for $n(\geq 0)$ system tasks at level 2
- m user task queues for $m(\geq 1)$ types of user tasks with different dispatching priorities within level 3.

Furthermore, a priority level indexed with i=0 is contained, which always is non-preemptive and for which no request must wait, since the user task phases on level 3 throughout are running enabled with respect to I/O-interrupt requests (i.e. they always can be interrupted).
So, levels 0,1 and 2 represent the processing of operating system code in supervisor state, whereas level 3 is the level for the processing of user task phases in problem state.

In the following, the operating system parts are described.

The processor model is considered on the level of user task phases defined above and is driven by external I/O-interrupt requests, arriving with rate λ_1 from the I/O subsystem. These requests represent messages from the channels and devices which have to be investigated and handled by processing appropriate I/O-interrupt handling routines.

In most cases, the I/O interrupt is signalizing the end of a data transfer between main and secondary storage (channel end, device end). The consequence of such a message is first an attempt to issue the next START-I/O, i.e. to reschedule the same device or channel. After this has been done, the associated user task of the I/O interrupt is dispatchable again, i.e. is selectable for the beginning of its next user task phase. This is represented by the branch from level 1 into the associated queue of level 3.

There might be further reasons for I/O interrupts e.g. 'SEEK-end' of a disk, which does not result in making the associated user task dispatchable again. This is represented by a branch leaving the processor after level 1. In the model this path is selected by an arbitrary fraction p_{10} of I/O-interrupt requests. This fraction e.g. is 0 if for each physical I/O request only 1 interrupt is needed. Consider e.g. disks with RPS (rotational position sensing, see also e.g. /5/), where normally no separate I/O interrupt for a seek end is necessary since the I/O can be performed with one channel program.
Thus the total pathlengh of level 1 service may be interpreted to include

- a part for the analysis of the type of interrupt (a general entry routine)
- the proper interrupt handling routine
- the channel device scheduling
- the dispatcher performing a task switch.

Level 3 of the model consists of $m(\geqslant 1)$ different queues for m different types of user tasks with different user priorities. This means, if a user task phase of a user task of the last priority m has been interrupted by an I/O interrupt making a class 1 user task dispatchable, this class 1 user task is dispatched before resuming the interrupted user task phase of a class m user task. When the end of a user task phase is reached, an immediate switch into supervisor state is performed (e.g. by a supervisor call SVC) to prepare the I/O access. This includes an entry routine, the construction of appropriate channel programs , queue entries, channel scheduling and the dispatching of the next task. There are further activities necessary, depending on the type of I/O. This may be in case of user -I/O the translation of the channel program (CCW Translation) or in case of a page fault the execution of the page replacement algorithm.

In the model all these activities are represented by a general non-preemptive operating system part at level 0 and by n different system tasks at level 2.

"System tasks" here are considered to be internal tasks of an operating system combining a certain amount of work for a special purpose.This may be e.g. a page manager which is responsible for the paging,or may be a task for the fetching of transient parts of the operating system.

If in case of an I/O access such a system task is needed, then during the non-preemptive operating system phase with index 0 the queue entry for a system task is prepared, performed and analyzed by the dispatcher. If no system task is included, requests leave the processor in the branch labeled with the associated fraction p_{30}.

As can be seen, in the terminology used, operating system phases correspond to levels 0, 1 and 2, whereas only in level 2 system tasks are processed.

2.3 Processor Scheduling

To completely define the functioning of the model, it is necessary to describe the interrupt scheme (i.e. which service may be interrupted by I/O interrupt requests) and the procedure for the scheduling of waiting requests (dispatching rule).

INTERRUPT SCHEME:

The only requests arriving from outside of the processor (i.e. being asynchronous) are the I/O-interrupt requests with index 1. Table 1 shows, whether a class i service may be interrupted by an arriving I/O-interrupt request.

level	0	1	2	3
status	n	n	p if $c_{12}=0$ n if $c_{12}=1$	p

Table 1: Interrupt Scheme (p=preemptive resume,n=non-preemptive, c_{12}= control variable)

Note that class 0 service cannot be interrupted at all, i.e. class 0 service is running disabled for all I/O-interrupt requests. Also, I/O interrupt requests do not interrupt themselves. For ease of notation a control variable c_{12} has been introduced, indicating purely preemptive system tasks ($c_{12}=0$) or purely non-preemptive system tasks ($c_{12}=1$). These two bordering cases for level 2 have been included, since often operating system phases are a sequence of preemptive and non-preemptive parts of code, implemented e.g. by ENABLE and DISABLE commands. Also interruptions might be allowed only at certain predetermined points.

Naturally all user code (running in problem state) is preemptive.

DISPATCHING RULE:

This rule determines which request of the waiting requests will get the control on the processor (will be dispatched) if a service has reached a normal end.

If there are requests waiting for processor service, a request with momentary highest dispatching priority (lowest priority index) is chosen for service. Note that level 3 might be considered to consist of m separate interrupt levels (i.e. class 3_1 always is dispatched before an interrupted class 3_2 request).
Within each single queue the queue discipline FIFO is adopted.

2.4 Definitions and Assumptions

The model consists of levels 0,1,2 and 3. In addition, level 2 is composed of $n(\geqslant 0)$ sublevels where n is the number of system tasks. Also level 3 is formed by $m(\geqslant 1)$ sublevels with m as the number of different types of user tasks. Let these levels and sublevels be denoted by i ($i=0,1,2,3,2_1 \ldots 2_n, 3_1 \ldots 3_m$).

Now, the total flow or response time for a request from its arrival at the I/O-interrupt queue and the time being completely served by the processor (i.e. the time instant leaving the processor) is the sum of the flow times in the different levels involved. The flow time T_{Fi} for a priority level i is

$$T_{Fi} = T_{Wi} + T_{WSi} + T_{Hi} \qquad (1)$$

T_{wi} is the initial waiting time for a level i, i.e. the waiting time from the arrival up to the first period of service. If level i is interruptible, the service time T_{Hi} may not be given continuously due to interrupts. These interrupts induce a total subsequent waiting time T_{WSi}. The sum of service time and subsequent waiting time often is called residence time /7/.

Let each level or sublevel i be characterized by

λ_i = arrival rate of requests
$h_i = E(T_{Hi})$ mean service time
$A_i = \lambda_i \cdot h_i$ offered traffic (utilization) for level i (2)

The distribution functions of all service times may be arbitrary with

c_{Hi} = coefficient of variation (standard deviation/mean value).

It is assumed throughout that the arrival process of class 1 is a Poisson process and that the service times are independent of each other.

Furthermore, probabilistic branching is assumed. The associated branching probabilities be p_{ij}. The probability of leaving the processor after the class 1 service is p_{10}, whereas with probability p_{30} no system task is involved.

Naturally,

$$\sum_{j=1}^{n} p_{2j} = 1 \quad \text{and} \quad \sum_{j=1}^{m} p_{3j} = 1 \,.$$

It is very simple to calculate all arrival rates λ_i and all utilizations A_i for the model.

For ease of notation, let

$$A_2 = \sum_{j=1}^{n} A_{2j} \quad , \quad A_3 = \sum_{j=1}^{m} A_{3j} \qquad (3a,b)$$

$$z_i = \frac{1+c_{Hi}^2}{2} \cdot h_i \cdot A_i = \frac{\lambda_i}{2} \cdot E(T_{Hi}^2) \qquad (4)$$

$$h_o' = h_o + (1-p_{30}) \cdot \sum_{j=1}^{n} p_{2j} \cdot h_{2j} \qquad (5)$$

$$A'_{3j} = \lambda_{3j} \cdot (h_{3j} + h'_0) = \lambda_1 \cdot (1 - p_{10}) \cdot p_{3j} \cdot (h_{3j} + h'_0) \qquad (6)$$

$$j = 1 \ldots m$$

2.5 Related Models

Priority systems with feedbacks have been investigated e.g. by ENNS /8/. Also the well-known time-sharing models belong to this type of queueing systems /11/.

A processor model with explicit operating system phases has been considered by LEWIS,SHEDLER /14/ as part of a cyclic queueing system. That closed queueing model has been treated by Imbedded Markov Chain analysis,assuming a finite number of identical jobs with same priority.

Recently,HERZOG /10/ investigated flexible priority models for communication processors serving different classes of jobs or requests and taking into account system overhead especially for interrupt handling.That processor model with so-called preemption-distance priorities (cf.e.g. /9/) is for the special case of pure preemptive priorities identical with a simplified version of the model treated here,having $p_{10}=0, p_{30}=1, h_0=0$ and no system tasks (n=0).

3 QUEUEING ANALYSIS

3.1 Method of Analysis

For the investigation of priority systems with feedbacks,several methods can be and have been used,which are sometimes more or less related.These methods include in many cases the consideration of certain requests,often called test-requests,during their flow through the system.From this,the moments of the random variables involved (waiting time,response time) are calculated.

This basic principle was used e.g. by HERZOG /9,10/ and applied to systems with preemption-distance priorities in order to calculate the first moment of the waiting or response times.

Furthermore,when considering test-requests it is tried to define partially equivalent M/G/1 systems /7,8,11/ in order to directly apply known results for this single server system without priorities,also for higher moments.

The method of analysis used for the special processor model treated in this paper,can be considered to be a congenial extension and application of COBHAM's calculation method /6/,which has been developed and applied to priority systems with non-preemptive priorities. As will be explained,this method mainly consists of two parts

- the calculation of total initial amounts of work
- the application of results of generalized busy periods for M/G/1 systems

When calculating different initial amounts of work,it was in part necessary to consider test-requests during part of their flow through the system as it was demonstrated by HERZOG.

As will be also shown in the following,special attention had to be given to the mechanism of interruptions.This means that at an arbitrary instant also interrupted requests are present,only requiring part of a service time to be completed.

3.1.1 Initial Waiting Times

The calculation of the mean initial waiting times $E(T_{Wi})$ of a request in queue i is performed in two steps:

- First, a so-called initial amount of work T_{Oi} is calculated. This is a certain backlog of processor work at that instant, the considered request of class i arrives at queue i.It is composed only by that services,which - according to the interrupt and dispatching disciplines - have to be done before the considered request might be dispatched for class i service.

This initial amount of work may be enlarged by external requests arriving later on.

- Therefore,secondly,the extension of the waiting time is calculated being induced by arriving external requests.

Since the external arrivals form a Poisson process,it can be easily shown that

$$E(T_{Wi}) = \frac{E(T_{Oi})}{1 - A_i^*} \qquad (7),$$

where A_i^* is a partial effective server utilization induced by external arrivals.This result,already applied by COBHAM /6/,can also be obtained by considering generalized busy periods in M/G/1 systems (cf. e.g./1,7,11/).

Due to the FIFO discipline within all queues,requests arriving later on in class 1 do not influence the initial waiting time of a considered class 1 request,and therefore

$$A_1^* = 0 \qquad (8)$$

All class 1 requests arriving during the initial waiting time of a request in any class 2 queue will be dispatched previously,but only for receiving service in class 1.So

$$\Lambda_{2j}^* = \Lambda_1 \qquad j = 1 \ldots n \qquad (9)$$

Regarding a request waiting in queue 3_i, it must be taken into account that all class 1 requests are dispatched previously and that some of them are continued in a higher priority class of level 3 and then possibly in class 2:

$$A_{3i}^* = A_1 + \sum_{j=1}^{i-1} A'_{3j} \qquad (10)$$

The appropriate initial amounts of work will be calculated separately in chapters 3.2 ff.

3.1.2 Subsequent Waiting Times

The calculation of the subsequent waiting times for interruptible services analogously result in

$$E(T_{WSi}) + E(T_{Hi}) = \frac{E(T_{Hi})}{1 - A_i^{**}} \qquad (11a)$$

or

$$E(T_{WSi}) = \frac{A_i^{**}}{1 - A_i^{**}} \cdot E(T_{Hi}) \qquad (11b)$$

with A_i^{**} as another effective server utilization.

In this model

$$A_i^{**} = \begin{cases} 0 \text{ for non-interruptible service} \\ A_i^* \text{ else} \end{cases} \qquad (12)$$

Using the well-known Little's theorem,for each waiting time also corresponding queue lengths can be obtained.

3.2 Class 1 Waiting Time

The (initial) waiting time of class 1 requests in the I/O-interrupt queue is identical with the initial amount of work T_{01} for this queue.This backlog of work is composed of requests of different classes,each request needing a certain time or residual time to be served.Table 2 shows the mean or expected number of requests together with their service demand.

Mean number	A_0	A_1	$\lambda_1 \cdot E(T_{W1})$	$c_{12} \cdot A_{2\nu}$ $(\nu=1\ldots n)$
Mean (residual) time	$\frac{1+c_{HO}^2}{2} \cdot h_O$	$\frac{1+c_{H1}^2}{2} \cdot h_1$	h_1	$\frac{1+c_{H2\nu}^2}{2} \cdot h_{2\nu}$

Table 2: Components of the mean initial amount of work for class 1

If at an arbitrary instant a request of class 1 arrives,with probability A_i a class i service is in progress,having an expected residual service time of

$$\frac{1+ c_{Hi}^2}{2} \cdot h_i \qquad (i = 0,1).$$

This is a well-known result from renewal theory.

The mean number of waiting class 1 requests is $\lambda_1 \cdot E(T_{W1})$,each requiring a full class 1 service.If class 2 service cannot be interrupted, $(c_{12}=1)$,a further part has to be added.
So the mean waiting time in the I/O-interrupt queue turns out to be

$$E(T_{W1}) = \frac{z_O + z_1 + c_{12} \cdot \sum_{\nu=1}^{n} z_{2\nu}}{1 - A_1} \qquad (13)$$

3.3 Class 2 Waiting Times

To calculate the initial waiting time of a request in a queue 2_i, consideration of a test request may start when it begins service at level 0.Then no other requests are present in the queues of levels 1 or 2,since user task phases are always interruptible.

So the initial waiting time of a class 2_i request is identical with the sum of all class 1 service times of all requests arrived during the non-preemptive phase 0,possibly enlarged by further arriving class 1 requests.At the end of the service time T_{HO} of the considered request in the mean $\lambda_1 \cdot h_O$ class 1 requests have arrived, each claiming for a mean service time h_1.Therefore the initial amount of work is

$$E(T_{O2i}) = A_1 \cdot h_O$$

So

$$E(T_{W2i}) = \frac{A_1}{1 - A_1} \cdot h_O \qquad i=1...n \qquad (14)$$

The subsequent waiting time is

$$E(T_{WS2i}) = (1 - c_{12}) \cdot \frac{A_1}{1 - A_1} \cdot h_{2i} \qquad (15)$$

$$i=1...n$$

3.4 Class 3 Waiting Times

To calculate the initial waiting time of a class 3_i request,it is necessary to pursue such a request (cf. fig.2).

Let such a class 3_i request arrive at time t at queue 1 and after waiting and service time it arrives at time t' at queue 3_i.

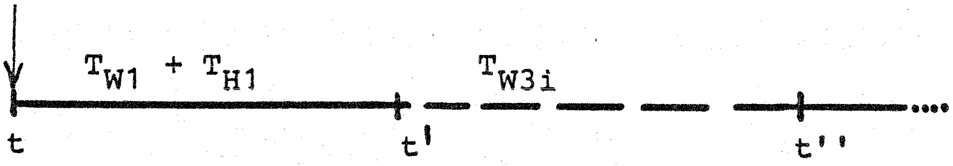

Fig 2: Time diagram for considered test request of class 3_i

The initial amount of work for this test request at time t' is composed of

- requests already present in the system at time t

and - requests arrived during t and t'

Table 3 is a survey on these different components of the initial amount of work $E(T_{O3i})$.

	Mean Number	Residual Mean Time	
1)	$\lambda_1\cdot E(T_{W1})+A_1$	$\sum_{\gamma=1}^{i}(1-p_{10})\cdot p_{3\gamma}\cdot(h_{3\gamma}+h_0^{\prime})$	
2)	$\lambda_{2\gamma}\cdot E(T_{W2\gamma})$	$h_{2\gamma}$	$\gamma=1...n$
3)	$\lambda_{2\gamma}\cdot E(T_{WS2\gamma})$	$\frac{1+c_{H2\gamma}^2}{2}\cdot h_{2\gamma}$	
4)	$A_{2\gamma}$	$(1-c_{12})\frac{1+c_{H2\gamma}^2}{2}\cdot h_{2\gamma}$	
5)	A_0	$(1-p_{30})\cdot\sum_{\gamma=1}^{n}p_{2\gamma}\cdot h_{2\gamma}$	
6)	$\lambda_{3\gamma}\cdot E(T_{W3\gamma})$	$h_{3\gamma}+h_0^{\prime}$	$\gamma=1...i$
7)	$\lambda_{3\gamma}\cdot E(T_{WS3\gamma})$	$\frac{1+c_{H3\gamma}^2}{2}h_{3\gamma}+h_0^{\prime}$	
8)	$\lambda_{3\gamma}h_{3\gamma}$		
9)	$\lambda_1\cdot(E(T_{W1})+h_1)$	$h_1+\sum_{\gamma=1}^{i-1}(1-p_{10})\cdot p_{3\gamma}\cdot(h_{3\gamma}+h_0^{\prime})$	

Table 3: Components of the mean initial amount of work $E(T_{O3i})$ for class 3_i

Lines 1 to 8 refer to requests being already at time t in the system, having different residual mean times.

Line 9 refers to requests arriving during t and t'.They all will receive class 1 service,whereas only classes 3_1 up to 3_{i-1} of them are receiving total service before the considered class 3_i request.

With

$$E(T_{W3i}) = \frac{E(T_{O3i})}{1 - A_{3i}^{*}}$$

after standard manipulations it is obtained

$$E(T_{W3i}) = \frac{1}{1-A_1-\sum\limits_{\gamma=1}^{i} A'_{3\gamma}} \cdot \Bigg[\sum_{\gamma=1}^{i-1} A'_{3\gamma} \cdot E(T_{W3\gamma}) + \ldots$$

$$\ldots + \left[E(T_{W1}) + h_1\right] \cdot \left\{ A_1 + A'_{3i} + 2 \cdot \sum_{\gamma=1}^{i-1} A'_{3\gamma} \right\} + \ldots \qquad (16)$$

$$+ \frac{1}{1-A_1} \cdot \left[A_1 A_2 h_o + (1-c_{12}) \cdot \sum_{\gamma=1}^{n} z_{2\gamma} \right] + \sum_{\gamma=1}^{i} \frac{z_{3\gamma} + A_{3\gamma} h'_o}{1-A_1-\sum\limits_{j=1}^{\gamma-1} A'_{3j}} \Bigg]$$

From this recursive formula for the mean initial waiting time in a queue 3_i (i=1...m), with the help of a homogeneous and a particular solution of the resulting inhomogeneous difference equation it is possible to derive an explicit solution

$$E(T_{W3i}) = \frac{(1-A_1) \cdot E(T_{F1}) + A_1 A_2 h_o + (1-c_{12}) \cdot \sum\limits_{\gamma=1}^{n} z_{2\gamma} + \sum\limits_{j=1}^{i} z_{3j} + h'_o \cdot \sum\limits_{j=1}^{i} A_{3j}}{\left[1-A_1-\sum\limits_{j=1}^{i-1} A'_{3j}\right] \cdot \left[1-A_1-\sum\limits_{j=1}^{i} A'_{3j}\right]} - E(T_{F1})$$

$$i=1\ldots m \qquad (17)$$

where

$$E(T_{F1}) = E(T_{W1}) + h_1 \qquad (18)$$

is the expected flow time of class 1 requests.

It has been already indicated that the subsequent waiting time of a class 3_i request (cf.(10),(11),(12)) is

$$E(T_{WS3i}) = \frac{A_1 + \sum\limits_{j=1}^{i-1} A'_{3j}}{1 - A_1 - \sum\limits_{j=1}^{i-1} A'_{3j}} \cdot h_{3i} \qquad (19)$$

3.5 Response Times

The response time for a request belonging to a user task of class i i.e. of user priority i (i=1...m) is the time from the arrival of the associated I/O interrupt up to the final release of the processor:

$$E(T_{Ri}) = E(T_{F1}) + E(T_{W3i}) + E(T_{WS3i}) + h_{3i} + h_o + \ldots \qquad (20)$$

$$\ldots + (1-p_{3o}) \cdot \sum_{j=1}^{n} p_{2j} \cdot \left[E(T_{W2j}) + E(T_{WS2j}) + h_{2j}\right]$$

The mean total time a job of user priority i is in the total computer system would be obtained by adding the mean time in the channel device subsystem per I/O access and by multiplication with the mean number of physical I/O accesses.

4 NUMERICAL RESULTS

4.1 Influence of System Overhead

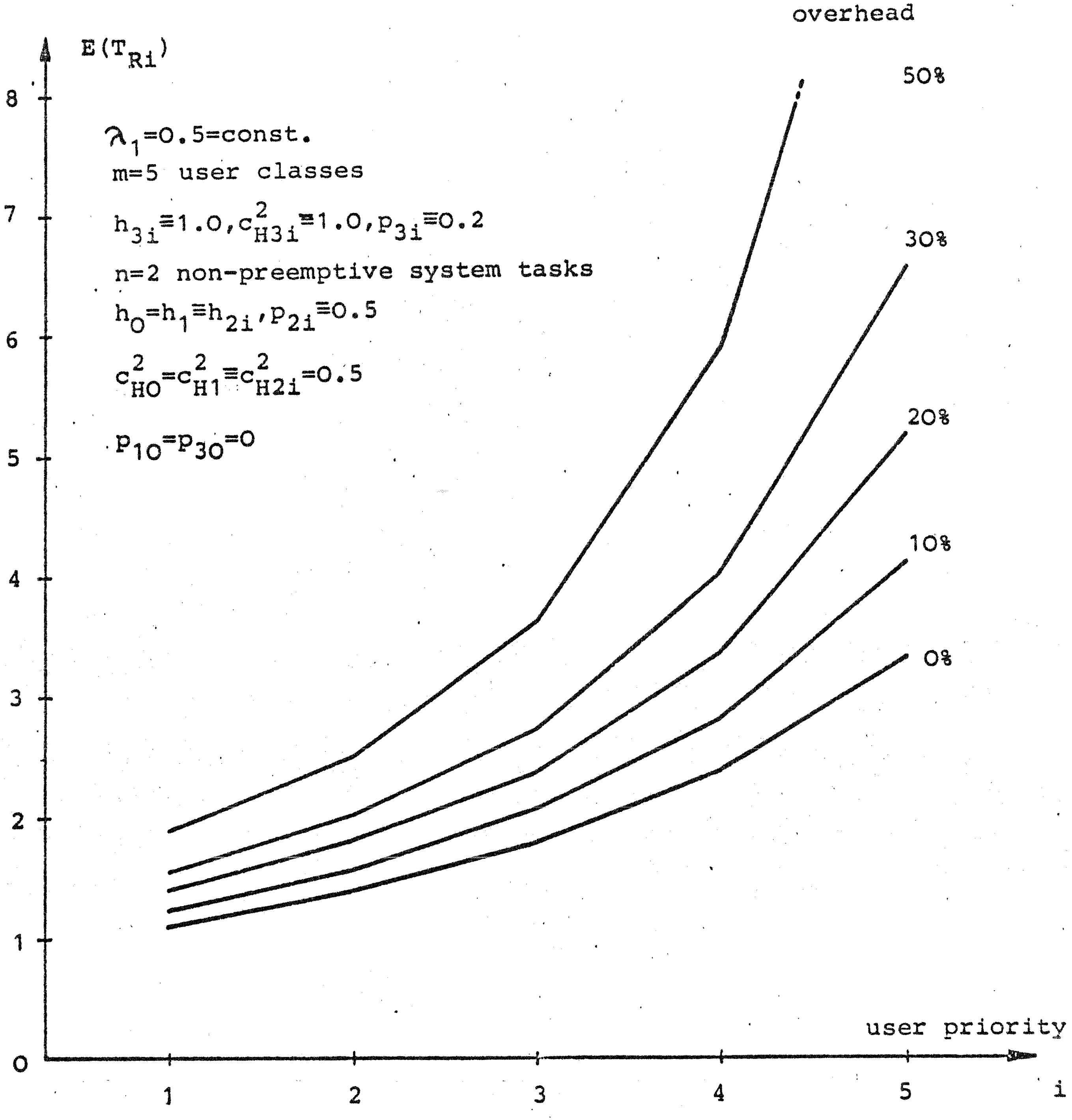

Fig 3: Influence of system overhead on the mean response times of different user classes

Fig.3 shows the influence of the operating system phases on the mean response times of m=5 different user priority classes.
The processor utilization for user task processing has been taken fixed

$$A_3 = \sum_{i=1}^{m} A_{3i} = 0.5$$

whereas the additional system overhead $A_0+A_1+A_2$ has been varied up to 0.25.

4.2 Waiting Times within Internal Operating System Queues

It has been confirmed by several calculations that the waiting times within internal operating system queues often can be neglected compared with the waiting times in a user task queue.
To give some figures,for the highest user priority the fraction of waiting in internal operating system queues (levels 1,2) was up to 10% for a system with 50% overhead and up to 20% for the same number of user and operating system instructions (100% overhead).

4.3 Comparison of Total Response Times

For many purposes it is sufficient to consider the processor as a black box and to describe its behaviour by the different response times for different user priorities.Of special interest here is the question,how large is the influence of the operating system phases on the discrimination of the different user priority classes.

Fig.4 shows the response times for a processor model with 3 user priorities as a function of the total processor utilization.

Compared with the calculation results are results for corresponding models without feedbacks.
If system overhead is considered to globally enlarge user task phases,a comparison model with 3 queues is obtained,each with a mean service time of h =1.4 and a squared coefficient of variation of 0.612.This model has been calculated for the two bordering cases of pure preemptive and pure non-preemptive services.

There is another possibility to calculate approximately the response times by assuming for each of the 6 queues in the example a Poisson input process.The resulting model with a mixture of preemptive and

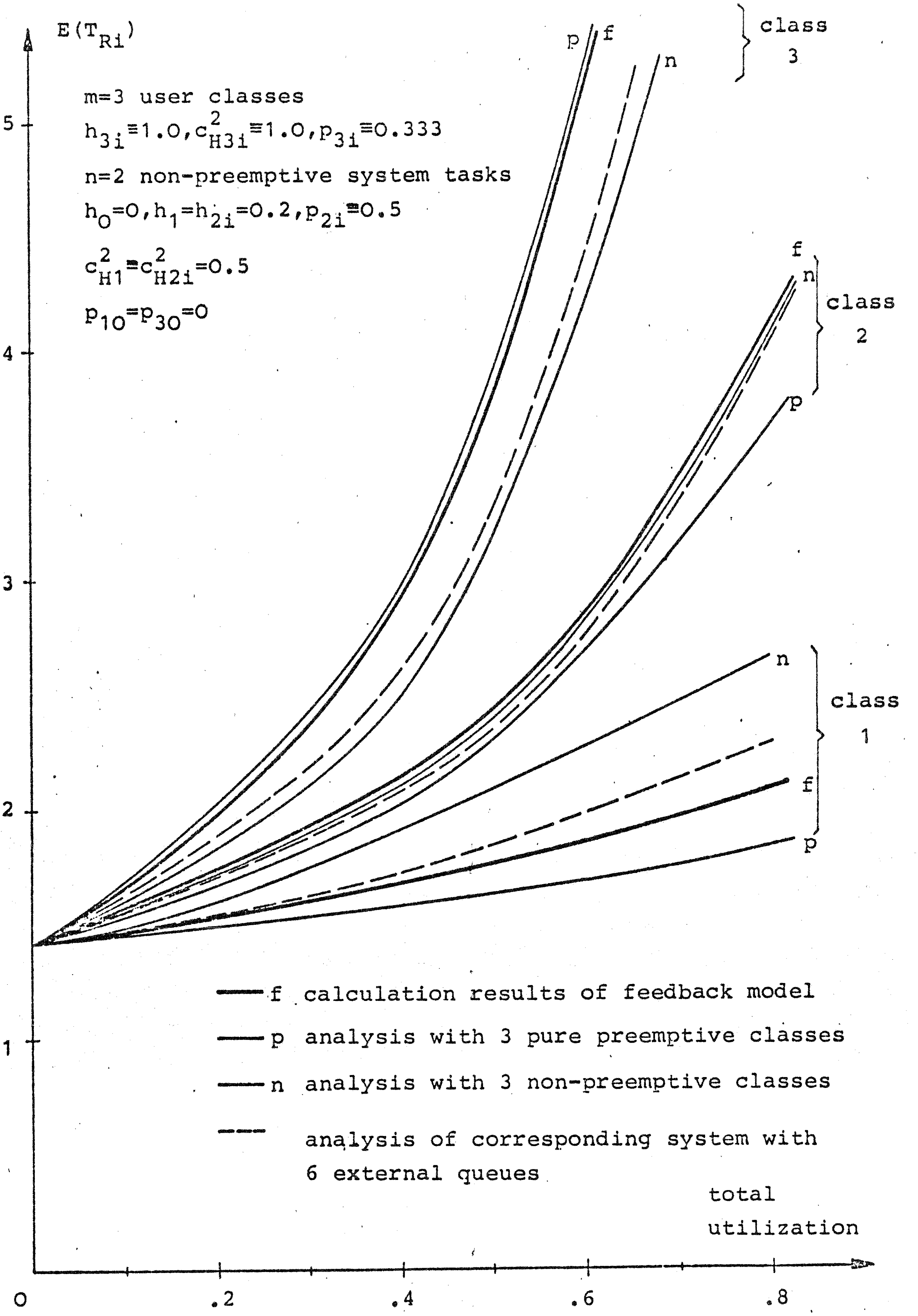

Fig. 4: Comparison of total response time results of the feedback model and of models without feedbacks

nonpreemptive classes has been calculated with the help of preemption-distance priority results /10/.

As can be seen, for this example none of the simpler models can be used with good accordance for the response times of all 3 user priority classes.

5 SUMMARY AND CONCLUSION

A processor model has been considered with different priorities of user jobs and with different operating system phases. The feedback model has been analyzed for general service time distribution functions, which only have to be specified by their first two moments.

Numerical examples have been given showing the influence of system overhead, and giving hints relating the necessity of modeling internal operating system queues. This necessity, as expected, heavily depends on the amount of overhead which can be selected properly in the model.

In any case, if waiting times within internal operating system queues are of interest, they have to be modeled.

The assumption of a Poisson input process of the I/O interrupts could be considered as a certain drawback of the analysis, since often models with a finite number of sources are more appropriate. Nevertheless, it can be expected that for same processor utilization and percentage of overhead the necessity for using a model with feedbacks is smaller for finite source than for this infinite source model.

Furthermore, when modeling distinct operating systems, much more complicated models may result, e.g. with

- feedbacks from the I/O-interrupt queue to system task queues
- I/O-compute overlap for the same job
- gating mechanisms e.g. to preserve integrity of control tables

Such a more detailed model has been developed and described in /12/ based on the supervisor of the operating system DOS/VS.

REFERENCES

/1/ ADIRI,I. Introduction to queueing theory with applications to computer systems
Computer Science Department Monograph Series,RA 42, IBM Research Division,Yorktown Heights,1972

/2/ ANDERSON,H. A.Jr. Approximating pre-emptive priority dispatching in a multiprogramming model.
IBM Journal of Res.Dev. 17(1973)4,533-539

/3/ BASKETT,F. CHANDY,K.M. MUNTZ,R.R. PALACIOS,F.G. Open,closed,and mixed networks of queues with different classes of customers
JACM 22(1975)2,248-260

/4/ BUZEN,J.P. Queueing network models of multiprogramming
Thesis,Harvard Univ. Camb. Mass.,1971

/5/ BUZEN,J.P. I/O subsystem architecture
Proc. IEEE 63(1975)6,871-879

/6/ COBHAM,A. Priority assignment in waiting line problems
Operations Research 2(1954),70-76

/7/ CONWAY,R.W. MAXWELL,W.L. MILLER,L.W. Theory of Scheduling
Addison-Wesley Publishing Company,1967

/8/ ENNS,E.G. Some waiting-time distributions for queues with multiple feedback and priorities
Operations Research 17(1969)3,519-525

/9/ HERZOG,U. Optimal Scheduling Strategies for Real-Time Computers
IBM J.Res.Dev. 19(1975)5,494-504

/10/ HERZOG,U. Priority models for communication processors including system overhead
8th International Teletraffic Congress (ITC), Melbourne,November 1976,Paper 623

/11/ KLEINROCK,L. Queueing systems,Vol I,II
John Wiley,1975/76

/12/ KRAEMER,W. Performance investigations with a DOS/VS based operating system model
Submitted to an IBM Journal

/13/ KUEHN,P. Zur optimalen Steuerung des Multiprogramminggrades in Rechnersystemen mit virtuellem Speicher und Paging
Lecture Notes in Computer Science,Vol 34,Springer, Berlin/NY 1975,567-580

/14/ LEWIS,P.A.W. SHEDLER,G.S. A cyclic-queue model of system overhead in multiprogrammed computer systems
JACM 18(1971)2,199-220

/15/ REISER,M. KOBAYASHI,M. Queueing networks with several closed subchains: theory and computational algorithms
IBM Research Rpt RC 4919,July 1974

/16/ REISER,M. Interactive modeling of computer systems
IBM Systems Journal 15(1976)4,309-327