U. Herzog

Optimal Scheduling Strategies for Real-Time Computers

Abstract: In order to fulfill response time constraints in real-time systems, demands are often handled by means of sophisticated scheduling strategies. This paper first shows how to describe and analyze arbitrary combinations of preemptive and non-preemptive (head-of-the-line) priority strategies and, second, presents an algorithm that yields the optimal priority strategy, taking into consideration constraints on the response time.

Introduction

Today there exists a wide variety of real-time computer systems. Depending on their applications, they range from small and simple systems to very complex configurations [1, 2]. In a typical example there may be several printers, disks, and tapes; the system may be connected to many interactive terminals, to process control devices, or to another real-time system, as indicated in Fig. 1. Accordingly, the number as well as the complexity of supervisory and application programs also varies over wide ranges. However, in efficient scheduling strategies, there are some typical features common to all real-time systems, as follows.

In order to fulfill response time constraints, urgent demands are often handled by means of an interrupt-driven operating system, i.e., by means of preemptive (hardware) priorities. On the other hand, there are less urgent demands that do not justify preemption. Sometimes even preemptive priorities are nonsensical, e.g., if the processor overhead for interruption is greater than the remaining processing time of the low priority demand. Therefore, most real-time computer systems serve the various demands by reasonable combinations of preemptive and non-preemptive (software) priorities [1–4]. Most theoretical investigations neglect this fact and deal either with pure preemptive or pure non-preemptive (head-of-the-line) priorities. Our objective is to explore theoretically those neglected aspects, hoping that practitioners will be able to modify existing models and develop new models based on this theory.

In this paper we first show how arbitrary combinations of preemptive and non-preemptive priorities can be uniformly described by means of the so-called "preemption distance." (It therefore seems reasonable to introduce the unifying term preemption-distance priorities.) Important performance values are determined for the system M/GE/1 with an arbitrary number of priority classes (GE = general Erlangian; cf. Appendix 1). Because distributions and mean values may also differ in different classes, arbitrary types of demands can be modeled and analyzed with any required accuracy.

Most often implemented are preemption-distance priorities with a fixed number of interrupt levels. Within one interrupt level demands may occur with different priorities. However, they do not interrupt each other (cf. Fig. 2). The development of optimal scheduling strategies is demonstrated in the section on optimal scheduling for the above important class of preemption-distance priorities (fixed number of interrupt levels). The optimal strategy is defined as the strategy that gives the minimal number of expected interrupts for a given number of classes, a given traffic intensity and type per class, and given response time constraints for each individual class.

An efficient algorithm has been developed which is based on the technique of branch-and-bound and implicit enumeration.

Description of preemption-distance priorities

• *Uniform preemption distance for all classes, preemptive priorities, and non-preemptive priorities*

A general class of priority strategies can be characterized by the so-called preemption distance ξ, the uniform distance between a priority class and the next priority class being interrupted. Table 1 illustrates this definition: Demands of class p ($p = 1, 2, \cdots, P$; class 1 being the most urgent), interrupt only demands of classes $(p + \xi)$ to P, but not the intermediate classes $(p + 1)$ to $(p + \xi - 1)$. On the other hand, demands of the considered class p can be interrupted by classes 1 to $(p - \xi)$, but not by classes $(p - \xi + 1)$ to $(p - 1)$. Figure 3 and Table 2 show an example for a uniform preemption distance $\xi = 3$.

It is easily seen that two well known special cases of preemption-distance priorities are included: $\xi = 1$, preemptive priorities; and $\xi = P$, non-preemptive (head-of-the-line) priorities.

• *Arbitrary, nonuniform preemption distance for each priority class*

The preemption distance $\xi(p)$ may be defined individually for each priority class p ($p = 1, 2, \cdots, P$). Then arbitrary combinations of preemptive and non-preemptive priorities are allowed. Although nonuniform representation and analysis are possible, the method of determining the solution is rather complex.

A much more elegant solution is to use a uniform preemption distance while introducing "empty" priority classes: Dummy classes (with null arrival rates) are interleaved between the actual ones. This trick allows us to generate all scheduling strategies of practical interest (the only two special cases known in the literature [5, 6] are included). Furthermore, it facilitates the investigation of their influence on the waiting process.

Example An efficient strategy with nonuniform preemption distance is used for the I/O control in electronic switching systems [4]: The priorities are to be controlled

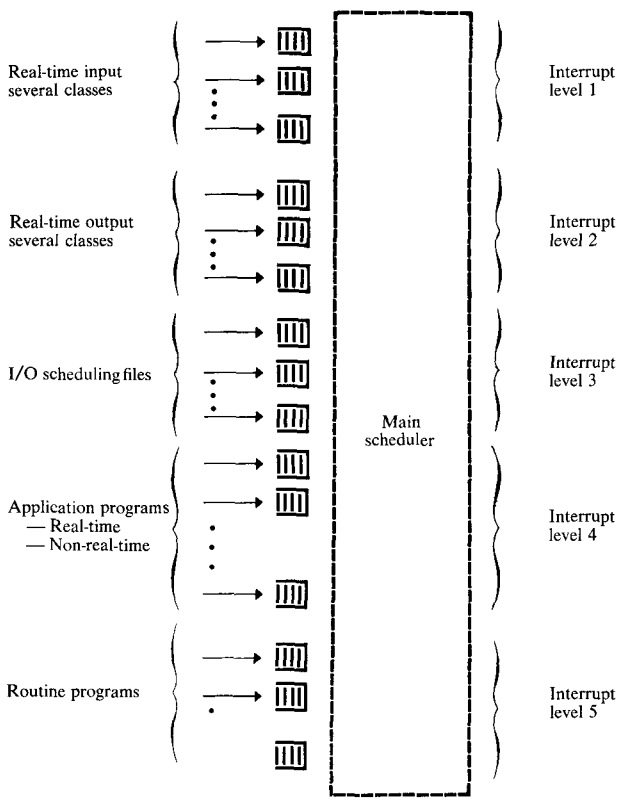


Figure 2 Queuing of the various demands in a real-time computer system.

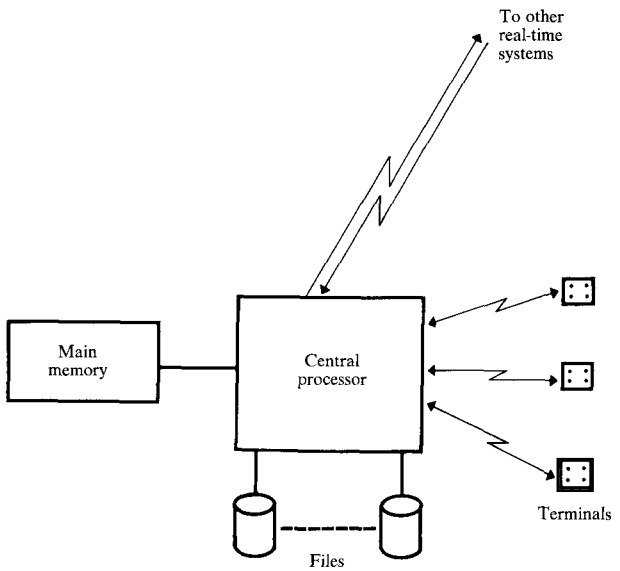


Figure 1 Small but typical configuration of a real-time computer system.

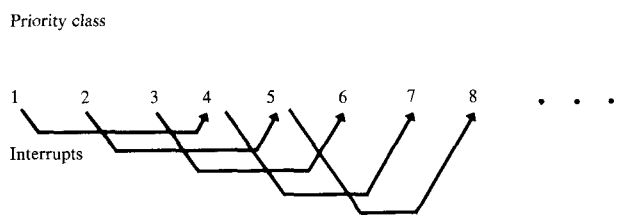
Table 1 Introduction of the preemption distance ξ. The special cases of preemptive priorities ($\xi = 1$) and non-preemptive priorities ($\xi = P$) are included.

Priority classes			
$1, 2, \cdots, p-\xi$	$p-\xi+1, \cdots, p-1$	$p, p+1, \cdots, p+\xi-1$	$p+\xi, \cdots, P$
demand interrupt class p	do not interrupt class p	not interrupted by class p	interrupted by class p

Table 2 Service mechanism and preemption distance (PD) corresponding to the priority strategy of Fig. 3.

Arriving demands of class	*Does not interrupt service of class*	*Interrupts service of class*	*Actual PD*
1	1, 2, 3	4, 5, 6, 7, 8, $\cdots$	3
2	1, 2, 3, 4	5, 6, 7, 8, $\cdots$	3
3	1, 2, 3, 4, 5	6, 7, 8, $\cdots$	3
4	1, 2, 3, 4, 5, 6	7, 8, $\cdots$	3
5	$\cdots$	$\cdots$	.
6	$\cdots$	$\cdots$	.
7	$\cdots$	$\cdots$	.

Figure 3 Example for a priority strategy with uniform preemption distance $\xi = 3$.



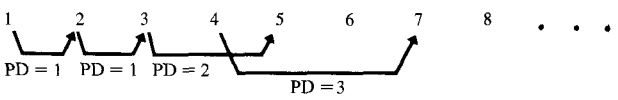


Figure 4 Second example for a combination of preemptive and non-preemptive priorities (cf. Table 3).

Table 3 Service mechanism and preemption distance (PD) corresponding to the priority strategy of Fig. 4.

Arriving demands of class	*Does not interrupt service of class*	*Interrupts service of class*	*Actual PD*
1	1	2, 3, 4, 5, 6, 7, 8, ···	1
2	1, 2	3, 4, 5, 6, 7, 8, ···	1
3	1, 2, 3, 4	5, 6, 7, 8, ···	2
4	1, 2, 3, 4, 5, 6	7, 8, ···	3
5	···	···	·
6	···	···	·
7	···	···	·

such that, e.g., priority class 3 interrupts demands of classes 5, 6, etc., but not the intermediate class 4, whereas class 4 interrupts only classes 7, 8, etc. (cf. Fig. 4 and Table 3). Table 4 illustrates how this interesting strategy can be interpreted and analyzed as a strategy with uniform preemption distance by introducing some appropriate dummy classes.

• *Fixed interrupt levels (two-dimensional representation)*

It was pointed out in the first section that preemption-distance priorities with a fixed interrupt level are most common in real-time computer systems (cf. Fig. 2). The optimization problem is to find a strategy that will guarantee a fast reaction of the system while minimizing additional overhead and hardware cost. To describe the algorithm that is applied to find optimal scheduling strategies, a two-dimensional notation is adopted here.

The notation is summarized in Table 5: Let G groups of priority classes be given. Demands of any group g ($g = 2, 3, \cdots, G$), are interrupted immediately when service for demands of more important groups is required (group 1 being the most urgent). Within a group g ($g = 1, 2, \cdots, G$), there exist ξ_g classes of demands that are of different priority; however, they do not interrupt each other (non-preemptive). For brevity, such a priority strategy is denoted by $F[\xi_1, \xi_2, \cdots, \xi_G]$ and each individual class by (g, γ). Obviously, the total number P of priority classes is given by the sum of all ξ_g.

Analysis

• *Arbitrary preemption-distance priorities*

The general solution for arbitrary combinations of preemptive and non-preemptive priorities characterized by a preemption distance has been obtained and presented in [4]. In that paper probabilities for waiting or for interrupts, mean waiting times, and other performance parameters are determined exactly for each class of demands. Arrival processes are assumed to be Poisson functions, and service times may have general Erlangian distributions, which may be different for different priority classes. The results are summarized in Appendix 1.

• *Preemption-distance priorities with fixed interrupt levels*

Traffic parameters and response time

Demands of each priority class (g, γ) are distributed, as in the general case, according to a Poisson process with mean arrival rate $\lambda(g, \gamma)$. The service time follows, individually for each class, a general Erlangian (GE) distribution with the mean value $b(g, \gamma)$. The mean response time $r_{g,\gamma}$ for each priority class (g, γ) of a specific priority strategy $F[\xi_1, \xi_2, \cdots, \xi_G]$ is given in Appendix 2.

Mean number of interrupts

Consider a specific class (g, γ) with the arrival rate $\lambda(g, \gamma)$ and the mean service time $b(g, \gamma)$. Then the overall arrival rate for demands that may interrupt this class is

$$\mu(g, \gamma) = \sum_{i=1}^{g-1} \sum_{j=1}^{\xi_i} \lambda(i, j).$$

The probability that such a demand occurs and interrupts a demand of class (g, γ) is $\rho(g, \gamma) = \lambda(g, \gamma) \cdot b(g, \gamma)$. Hence, the expected total number of interrupts occurring per time unit is given for priority strategy $F[\xi_1, \xi_2, \cdots, \xi_G]$ by

$$I[\xi_1, \xi_2, \cdots, \xi_G] = \sum_{g=2}^{G} \sum_{\gamma=1}^{\xi_g} \mu(g, \gamma) \cdot \rho(g, \gamma).$$

In the next section we show how to determine the strategy that generates the minimum expected number of interrupts.

Optimal scheduling

• *Formulation of the problem*

We want to find an optimal priority strategy, i.e., a strategy with the minimum expected number of interrupts, taking into consideration response time requirements individually for each priority class. The general problem is to minimize the objective function (expected number of interrupts) over all G and over all ξ_i:

$$I[\xi_1, \xi_2, \cdots, \xi_G] = \sum_{g=2}^{G} \sum_{\gamma=1}^{\xi_g} \rho(g, \gamma) \sum_{i=1}^{g-1} \sum_{j=1}^{\xi_i} \lambda(i, j),$$

subject to the response time constraints

$$r_{1,1}[\xi_1, \xi_2, \cdots, \xi_G] \leq r_{1,1},$$

$$r_{1,2}[\xi_1, \xi_2, \cdots, \xi_G] \leq r_{1,2},$$

$$\cdots,$$

$$r_{g,\gamma}[\xi_1, \xi_2, \cdots, \xi_G] \leq r_{g,\gamma},$$

$$\cdots,$$

$$r_{G,\xi_G}[\xi_1, \xi_2, \cdots, \xi_G] \leq r_{G,\xi_G},$$

where $G \geq 1, \xi_i \geq 1$.

- *Principle of solution*

In finding a method of solving the problem, several optimization methods have been investigated [7–9]. In comparison with classical optimization problems, this problem is characterized by the following features:

1. When starting our algorithm, there is no simple way to find a feasible solution. Therefore, the algorithm has to proceed with the dual objective of finding feasible solutions and of minimizing the number of interrupts.
2. The computing time for determining the actual response times for each priority class is much higher than the time required to generate a new strategy and determine the total number of interrupts.

An algorithm based on the branch-and-bound technique and implicit enumeration has been developed that takes into consideration the special features mentioned above. Roughly speaking, the main steps of the algotithm are as follows:

1. Determine a feasible solution and the corresponding number of interrupts to be used as an initial upper bound of the objective function.
2. Partition the set of all solutions into subsets. Determine a lower bound for all feasible solutions within a subset.
3. Those subsets whose lower bounds exceed the upper bound of the objective function are excluded. Check also conditions whether a subset may have any feasible solution at all.
4. Partition one of the remaining subsets further into several subsets, determine their lower bounds and exclude some subsets, etc.

- *Subsets, lower bounds, search for feasible solutions*

Definition of subsets

The general notation for a specific priority strategy $F[\xi_1, \xi_2, \cdots, \xi_G]$ was introduced in the subsection on fixed interrupt levels (two-dimensional representation) and Table 5. To find an efficient optimization procedure it is suitable to define sets of strategies.

Let S be the set of all feasible solutions $F[\xi_1, \xi_2, \cdots, \xi_G]$, i.e., the set of all possible priority strategies which satisfy the response time constraints. Furthermore, let $s[\nu_1, \nu_2, \cdots, \nu_f]$ be the subset of all feasible solutions for which the first f interrupt levels are fixed; additional interrupt levels, if any, can be arbitrary:

$$s[\nu_1, \nu_2, \cdots, \nu_f]$$
$$= \{F \mid F \in S \text{ and } \xi_1 = \nu_1, \xi_2 = \nu_2, \cdots, \xi_f = \nu_f\},$$

where $\Sigma_{i=1}^{f} \nu_i \leq P$.

Obviously subset $s[\nu_1, \nu_2, \cdots, \nu_f]$ contains all strategies $F[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}, \cdots, \nu_G]$. A small example is shown in Table 6 (see also Fig. 5).

Lower bounds for subsets

Let $I_L[\nu_1, \nu_2, \cdots, \nu_f]$ be the lower bound for all solutions of the subset $s[\nu_1, \nu_2, \cdots, \nu_f]$. Then the following observations are immediate and are therefore presented without formal proof:

Table 4 Generation of a nonuniform preemption distance (PD) by means of empty classes (classes with zero arrival rate: cf. Fig. 4 and Table 3).

Uniform PD	*Actual and empty class*	*Mean arrival rate*	*Actual class*	*Interrupt service of classes*	*Actual PD*
3	1	λ_1	1	4, 7, 9, 10, 11, 12, $\cdots$	1
3	2	0	·	$\cdots$	·
3	3	0	·	$\cdots$	·
3	4	λ_4	4	7, 9, 10, 11, 12, $\cdots$	1
3	5	0	·	$\cdots$	·
3	6	0	·	$\cdots$	·
3	7	λ_7	7	10, 11, 12, $\cdots$	2
3	8	0	·	$\cdots$	·
3	9	λ_9	9	12, $\cdots$	3
3	10	λ_{10}	10	$\cdots$	·
3	11	λ_{11}	11	$\cdots$	·
3	12	λ_{12}	12	$\cdots$	·

Table 5 Notation for priority strategies with fixed interrupt level (for an example, cf. Table 6).

Priority strategy $F[\xi_1, \xi_2, \cdots, \xi_G]$		
$\underbrace{(1,1)\cdots(1,\xi_1)}\cdots\underbrace{(g,1)\cdots(g,\gamma)\cdots(g,\xi_g)}\cdots\underbrace{(G,1)\cdots(G,\xi_G)}$		
Group 1	*Group g*	*Group G*
$g = 1, 2, \cdots, G,$	$G \leq P,$	$\sum_{g=1}^{G} \xi_g = P.$
$\gamma = 1, 2, \cdots, \xi_g,$	$\xi_g \leq P,$	

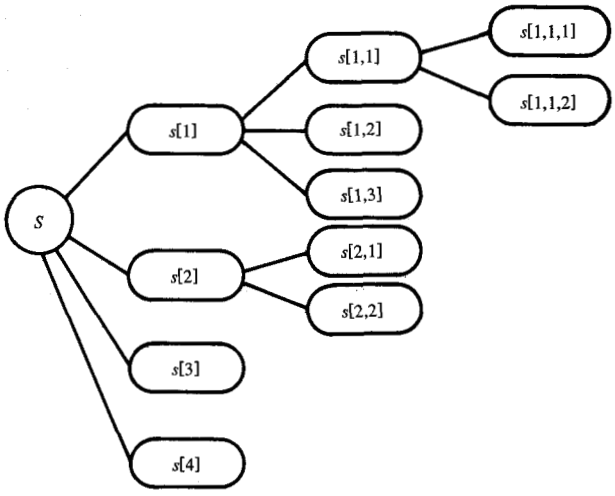


Figure 5 Sets of all feasible and infeasible solutions for $P = 4$ priority classes (i.e., all possible combinations of preemptive and non-preemptive priority strategies with fixed interrupt level; cf. Table 6).

1. A lower bound for the objective function of subset $s[\nu_1, \nu_2, \cdots, \nu_f]$ is given by

$$I_L[\nu_1, \nu_2, \cdots, \nu_f] = I[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}]$$
$$= \sum_{g=2}^{f+1} \sum_{\gamma=1}^{\nu_g} \rho(g, \gamma) \sum_{i=1}^{g-1} \sum_{j=1}^{\nu_i} \lambda(i, j),$$

where $\nu_{f+1} = P - \sum_{i=1}^{f} \nu_i$;

i.e., all ν_{f+1} remaining priority classes are lumped together into interrupt level $(f+1)$. All solutions with additional interrupt levels $(f+2, \cdots)$ cause more interrupts per time unit.

2. Per the definition of subset $s[\nu_1, \nu_2, \cdots, \nu_f]$, it includes all subsets $s[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}]$, $\nu_{f+1} \geq 1$. From 1 it follows directly that

$$I_L[\nu_1, \nu_2, \cdots, \nu_f] \leq I_L[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}].$$

3. If the lower bound $I_L[\nu_1, \nu_2, \cdots, \nu_f]$ corresponds to a feasible solution $F[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}]$, this solution is the best strategy for all sets $s[\nu_1, \nu_2, \cdots, \nu_f]$, $s[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}]$, $s[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}, \nu_{f+2}]$, etc., because $I_L[\nu_1, \nu_2, \cdots, \nu_f]$ is monotonically increasing with f.

Search for feasible solutions

Priority strategy $F[\nu_1, \nu_2, \cdots, \nu_{f-2}, \nu_{f-1}, \nu_f]$ is the lower bound of subset $s[\nu_1, \nu_2, \cdots, \nu_{f-1}]$; it contains f interrupt levels. Let $p = \sum_{i=1}^{f-1} \nu_i$. Obviously priority class p is within ν_{f-1}, i.e., p is in the group with lowest priority to cause an interrupt.

1. Suppose such a strategy $F[\nu_1, \nu_2, \cdots, \nu_{f-1}, \nu_f]$ is infeasible because of response time constraints for one or several classes i, $i \leq p$. Then any strategy in the subsets $s[\nu_1, \nu_2, \cdots, \nu_{f-2}, \nu_{f-1} + k]$, where

$$0 \leq k \leq P - \sum_{j=1}^{f-1} \nu_j,$$

cannot be a feasible solution because it gives at best the same response time for all classes i.

Table 6 Possible combinations of preemptive and non-preemptive priority strategies with fixed interrupt level and $P = 4$ classes as well as corresponding sets and subsets (cf. Fig. 5).

Possible priority strategies	*Sets and subsets*											
	S	$s[1]$	$s[2]$	$s[3]$	$s[4]$	$s[1,1]$	$s[1,2]$	$s[1,3]$	$s[2,1]$	$s[2,2]$	$s[1,1,1]$	$s[1,1,2]$
(1, 1)(2, 1)(2, 2)(2, 3)	x	x						x				
(1, 1)(1, 2)(2, 1)(2, 2)	x		x							x		
(1, 1)(1, 2)(1, 3)(2, 1)	x			x								
(1, 1)(1, 2)(1, 3)(1, 4)	x				x							
(1, 1)(2, 1)(3, 1)(3, 2)	x	x				x						x
(1, 1)(2, 1)(2, 2)(3, 1)	x	x					x					
(1, 1)(1, 2)(2, 1)(3, 1)	x		x						x			
(1, 1)(2, 1)(3, 1)(4, 1)	x	x				x					x	

2. Suppose strategy $F[\nu_1, \nu_2, \cdots, \nu_{f-1}, \nu_f]$ is not feasible because of the response time constraint for some class i $(i = p+1, \cdots, P)$. Then it is simpler to choose in the next step a strategy $F[\nu_1, \nu_2, \cdots, \nu_{f-1}, \nu'_f, \nu'_{f+1}]$ from the same subset $s[\nu_1, \nu_2, \cdots, \nu_{f-2}, \nu_{f-1}]$ rather than a strategy $F[\nu_1, \nu_2, \cdots, \nu_{f-2}, \nu''_{f-1}, \nu''_f]$ from some subset $s[\nu_1, \nu_2, \cdots, \nu_{f-2}, \nu''_{f-1}]$. This is so because the response times of the first p priority classes are not affected by the additional interrupt levels. Therefore, these response times need not be recomputed.

• *The algorithm*

The principle of the algorithm and some important properties of scheduling strategies and subsets are described in the previous subsections of this section. To illustrate the search for the optimal strategy, the general steps of the algorithm are described next; in Fig. 6 we outline a typical example.

At first we have to find an initial feasible solution (in our example let it be $s[1, 1, 1]$). Then we will try to improve this solution in a computationally efficient way, i.e., we will try to exclude computations of subsets that are unnecessary for the algorithm.

Denote for clarity the strategy that gives the lower bound of a subset $s[\nu_1, \nu_2, \cdots, \nu_f]$ by $F_L\{s[\nu_1, \nu_2, \cdots, \nu_f]\}$. Recall that $p = \Sigma_{i=1}^{f-1} \nu_i$. Then the general steps of the algorithm can be described as follows:

Step 1 Initialization

Start initially with set S; i.e., put $S = s[0]$. Set the upper bound of the objective function $I_\mu = \infty$.

Step 2 Feasibility

Check whether the considered set $s[\nu_1, \nu_2, \cdots, \nu_f]$ contains a feasible or infeasible strategy $F_L\{s[\nu_1, \nu_2, \cdots, \nu_f]\}$.

- If it is not feasible because of response time constraints for some class i, $i > (p + \nu_f)$, go to step 3.
- If it is not feasible because of response time constraints for some class i, $i \leq (p + \nu_f)$, go to step 4.
- Otherwise the solution is feasible; go to step 5.

Step 3 Splitting

The considered set may contain some feasible solutions. Therefore, split it into subsets $s[\nu_1, \nu_2, \cdots, \nu_f, \nu_{f+1}]$ and consider the next subset $s[\nu_1, \nu_2, \cdots, \nu_f, 1]$. Go to step 2.

(This step corresponds to a "horizontal search toward the leaves of the tree"; cf. Fig. 6, e.g., set S and subsets $s[1]$, $s[1, 1]$, $s[2]$.)

Step 4 New subsets of higher order

In this case, all subsets $s[\nu_1, \nu_2, \cdots, \nu_f + k]$, $k \geq 0$ cannot contain any feasible solution. Therefore, go back to the next "higher" level of subsets, if any, and investigate subset $s[\nu_1, \nu_2, \cdots, \nu_{f-2}, \nu_{f-1} + 1]$. Go to step 2. Otherwise go to step 6.

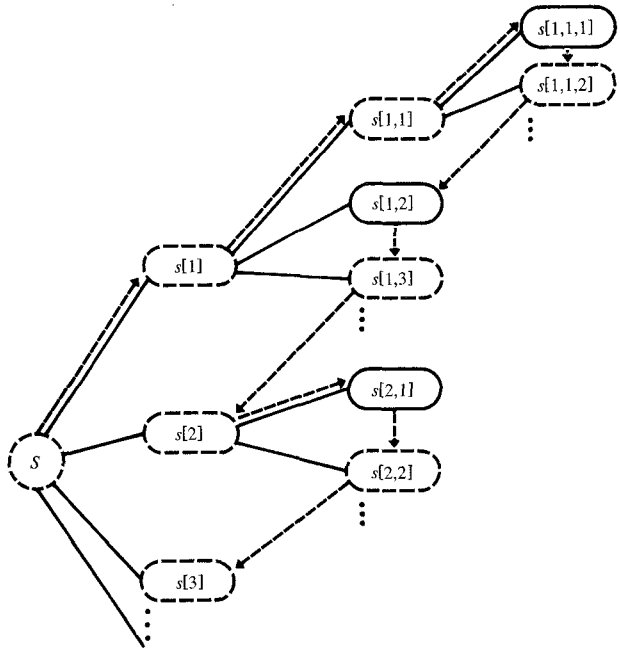
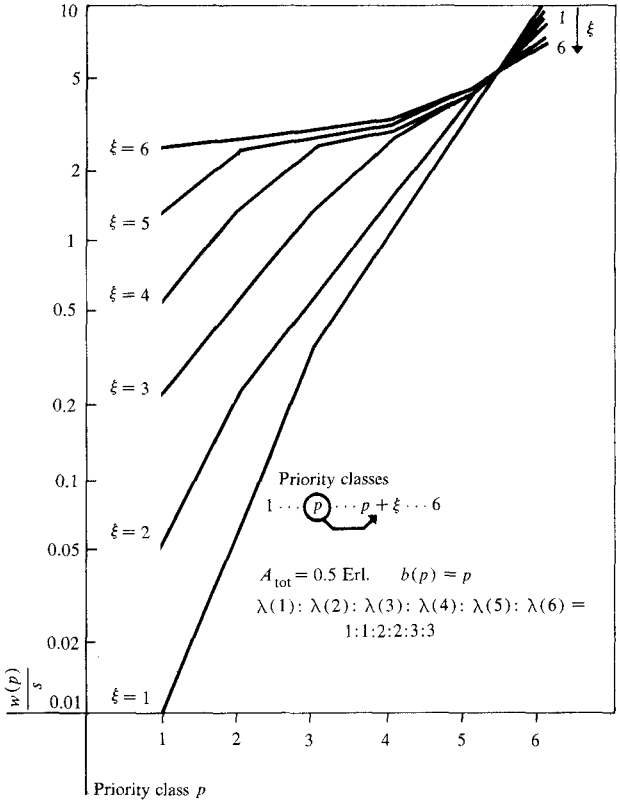


Figure 6 Search for the optimal strategy.

Figure 7 Mean waiting time $W(p)$. For this example all service times are assumed to be exponentially distributed, however, with different mean values per class.



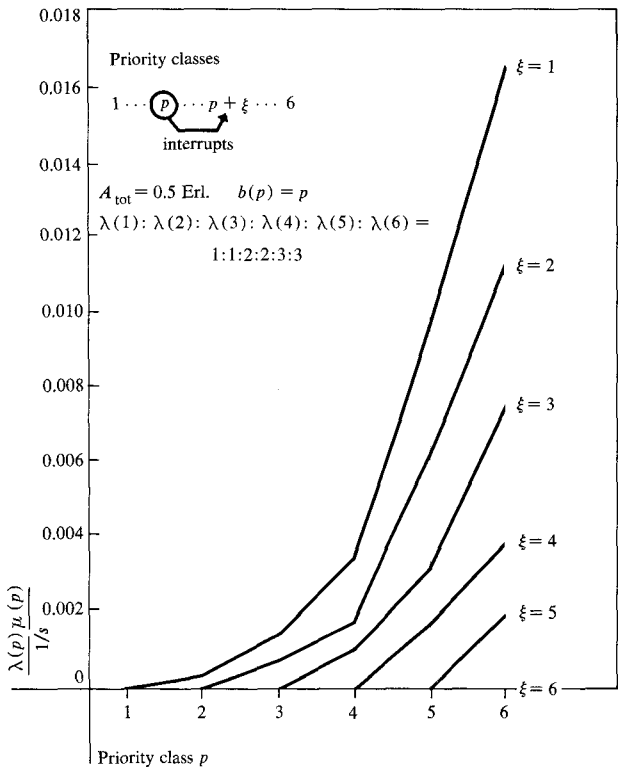


Figure 8 Mean number of interrupts per second for uniform preemption distance (cf. Fig. 7).

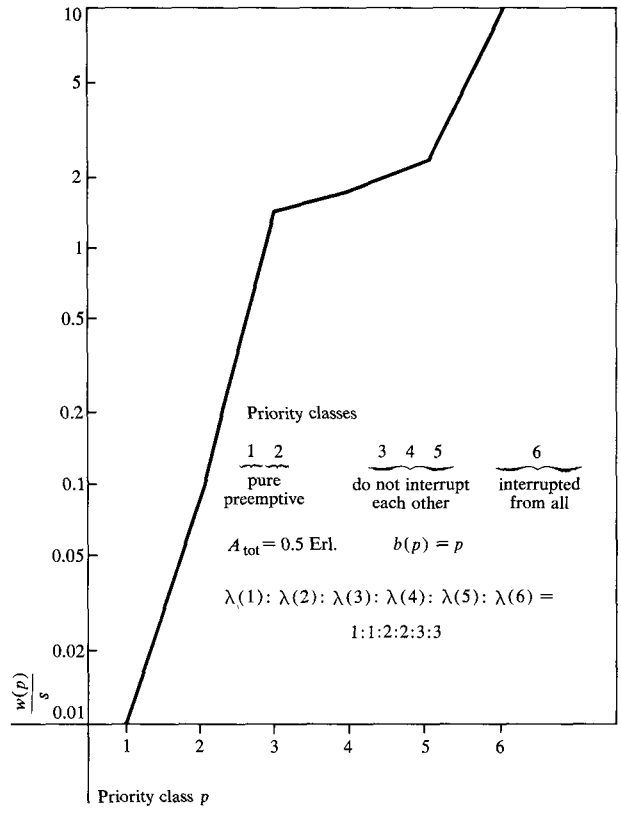


Figure 9 Typical example for an efficient combination of preemptive and non-preemptive priorities (compare with the priority strategies of Fig. 7).

(Typical examples of this step are the sets $s[1, 1, 2]$, $s[1, 3]$, and $s[2, 2]$ in Fig. 6).

Step 5 Feasible solutions
Check whether $I_L[\nu_1, \nu_2, \cdots, \nu_f]$ is less than the upper bound I_μ of the objective function. If so, set $I_\mu = I_L[\nu_1, \nu_2, \cdots, \nu_f]$. Consider next subset $s[\nu_1, \nu_2, \cdots, \nu_f + 1]$, if any, i.e., search "vertically" for better solutions in the tree. Go to step 2. Otherwise go to step 6.

(Figure 6 shows three examples of this type; subsets $s[1, 1, 1]$, $s[1, 2]$, and $s[2, 1]$.)

Step 6 Optimal solution
The I_μ of the objective function corresponds to the optimal scheduling strategy. Stop.

Numerical results

The following three examples show how various kinds of scheduling strategies and service times can be described, analyzed, and optimized in a uniform fashion. These examples also show the advantages of preemption-distance priority strategies as compared to pure preemptive or pure non-preemptive (head-of-the-line) priorities.

• *Uniform preemption distance*
Figure 7 shows the influence of the preemption distance on the mean waiting time. Traffic intensity and traffic character are constant.

The response time is 250 times smaller for $\xi = 1$ than for $\xi = 6$ (head-of-the-line). However, the price to be paid is shown in Fig. 8: An immense number of interruptions occur.

• *Nonuniform preemption distance*
Figure 9 shows a reasonable and often used "mixed" strategy between the two extremes (preemption-distance priorities with fixed interrupt level) resulting for the urgent demands exactly in the same fast response time as preemptive priorities, saving, however, a remarkable number of interrupts (cf. Fig. 10).

• *Extreme distribution functions for the processing times*
Figure 11 demonstrates which extreme types of distribution function are included in the solution presented above. Figure 12 shows for this example some values for the probability of waiting.

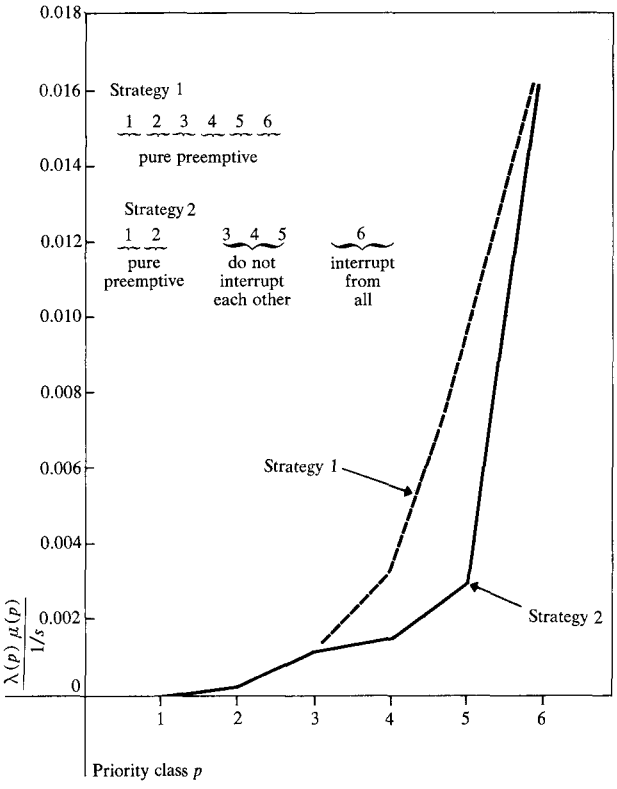


Figure 10 Mean number of interrupts per second for nonuniform preemption distance (cf. Fig. 9).

Priority class	Does not interrupt	Interrupts class	Preemption distance
1	1	2, 3	1
2	1, 2, 3	—	2
3	1, 2, 3	—	—

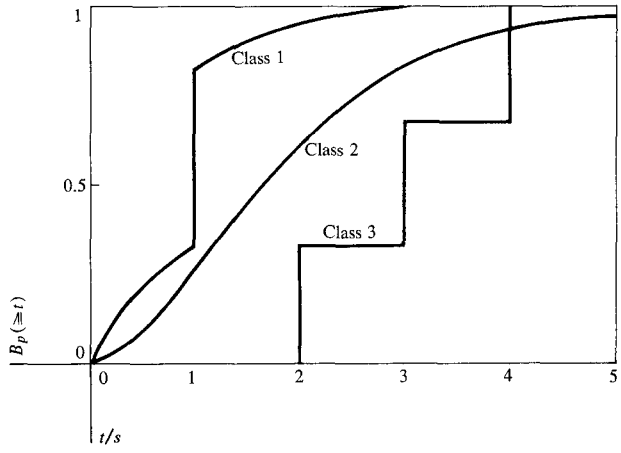
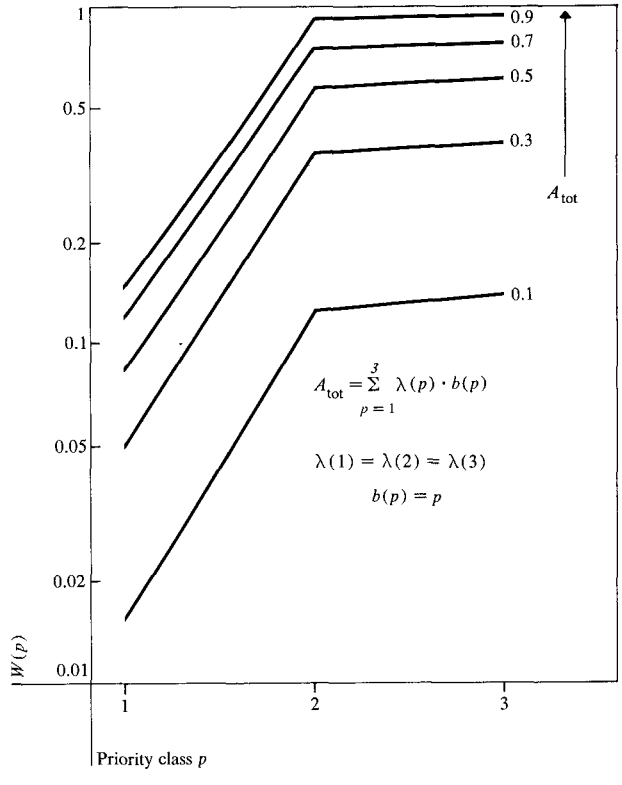


Figure 11 Example for three different types of service time distribution functions included in the general Erlangian distribution. The mean values are assumed to be proportional to the class number [$b(1) = 1$s, $b(2) = 2$s, $b(3) = 3$s]; the preemption distance is nonuniform.

Figure 12 Probability of waiting $W(p)$ for all three priority classes (priority strategy and service time distributions, cf. Fig. 11).



Summary and conclusions

Reasonable combinations of preemptive and non-preemptive (head-of-the-line) priorities are of major interest when operating real-time computer systems. They guarantee fast reaction to urgent signals, avoiding large overhead. All these strategies are uniformly described by introducing the preemption distance. The only two special cases known in the literature are included in the description and analysis as well.

The modeling with service times according to a general Erlangian distribution allows the accurate description of any type of processing time.

Mean values (response time, waiting time, ···) and characteristic probability values (probability of interruption···) show the main feature of a distinct strategy. For many practical applications these results are sufficient. A more detailed analysis is possible by means of the distribution function and a first step in this direction is the variance being taken into consideration.

Large real-time systems have duplexed register sets and hardware for interrupt handling. Therefore, the time for interrupt handling is very small compared to the processing times. However, designing small-system in-

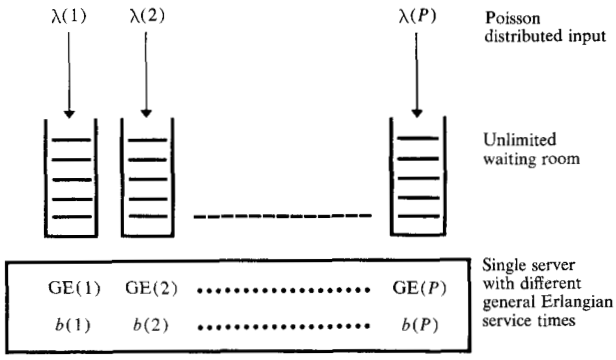


Figure A1 The investigated system.

terrupt handling may be done by software, adding a remarkable overhead. First results for these systems are already available.

Preemption-distance priorities with fixed interrupt levels are most important for practical applications. They guarantee a fast response to urgent requests while minimizing software overhead and hardware cost. Therefore, the optimization algorithm has been presented for this class of strategies. Obviously, it can be extended to arbitrary preemption-distance priorities. It can be also extended by introducing cost α_i per interrupt of level i; i.e., the varying importance of interrupts may be taken into consideration. Finally, the sensitivity analysis for various scheduling strategies is an interesting problem for future investigation.

Appendix 1: Analysis for arbitrary preemption distance

• *Structure and operating mode of the investigated system*

Figure A1 shows schematically the system to be investigated: Arriving demands are classified into P parallel queues according to their priorities. All queues are assumed to be unlimited, i.e., every arriving demand will be stored and processed. This assumption is almost always fulfilled, especially in systems with dynamic core allocation. All demands are served according to an arbitrary preemption-distance strategy, treated in the section on description of preemption-distance priorities; first-in, first-out is assumed within each priority class.

• *Traffic parameters*

Demands of each priority class p $(p = 1, 2, \cdots, P)$ are distributed according to a Poisson process with the mean arrival rate $\lambda(p)$:

$$A_p(\leq t) = 1 - e^{-\frac{t}{a(p)}} = 1 - e^{-\lambda(p)t}.$$

Service times follow, individually for each priority class, a general Erlangian (GE) distribution:

$$B_p(\leq t) = \sum_{\nu=1}^{l(p)} q_\nu(p) \times \left\{1 - e^{-\frac{t}{b_\nu(p)/k_\nu(p)}} \cdot \sum_{\eta=0}^{k_\nu(p)-1} \left[\frac{t}{b_\nu(p)/k_\nu(p)}\right]^\eta \Big/ \eta!\right\},$$

with the mean value

$$b(p) = \sum_{\nu=1}^{l(p)} q_\nu(p) \cdot b_\nu(p),$$

and the variance

$$\sigma^2(p) = \sum_{\nu=1}^{l(p)} \frac{k_\nu(p)+1}{k_\nu(p)} \cdot b_\nu(p)^2 \cdot q_\nu(p) - b(p)^2,$$

where

l: number of (fictitious) parallel "chains" of exponential "stages";
q_ν: probability that chain ν $(\nu = 1, 2, \cdots, l)$ is passed;
k_ν: number of stages for chain ν; and
b_ν: mean service time for one stage of chain ν.

It is worthwhile to note that this distribution allows us to approximate any type of distribution function of service times with any required accuracy. Obviously, it includes the hyperexponential $[k_\nu(p) = 1]$ as well as the Erlangian distribution $[l(p) = 1]$, both most important for many applications.

The time to handle interrupts is neglected. This assumption is also allowed because of large real-time computers that have duplexed register sets and hardware for interrupt handling. (For small systems without these facilities, see the summary and conclusions.)

• *Analysis*

General remarks

The most famous methods to investigate the stochastic behavior of such non-Markovian queuing systems are the method of embedded Markov chains [10], the phase method [11], the integral method [12], and the method of supplementary variables [13].

When arbitrary kinds of preemption-distance priorities were investigated, all methods failed because of the complex interdependencies among different priority classes. However, a general solution was possible by means of the method of moments: The fate of an individual demand of priority class p is pursued from its arrival up to the point where it leaves the system. All possibilities of interruption–processing, pushing back in the queue, etc.–are considered. Finally, when expectation values are introduced the presented solution can be obtained.

Characteristic performance values

The expected response time (time spent in the system, waiting and being processed) $r(p)$ for a demand of priority class p $(p = 1, 2, \cdots, P)$ is composed of the following five terms:

1. The expectation b_R $(\leq p + \xi - 1)$ of the remaining rest-service time for demands of the priority classes 1 to $(p + \xi - 1)$ that are present at its time of arrival in the server and will not be interrupted by the considered p-demand.
2. The expected time $w_I(p)$ necessary to serve demands of the priority classes 1 to p waiting in the system at its time of arrival.
3. Its expected time in service, $b(p)$.
4. The expected time $w_{II}(p)$ necessary to serve demands of preemptive priority classes 1 to $(p - \xi)$ which enter the system while the considered p-demand is still in the system.
5. The expected time $w_{III}(p)$ necessary to serve demands of the non-preemptive priority classes $(p - \xi + 1)$ to $(p - 1)$ which enter the system while the considered p-demand is still in the system, before its last interruption, however.

A detailed study of these five terms, presented in [14], leads to the following recursive solution for the expected response time for priority class p $(p = 1, 2, \cdots, P)$:

$$r(p) = \Bigg\{ \sum_{i=1}^{p-1} A(i) \cdot r(i) + b(p) + b_R(\leq p + \xi - 1) - \sum_{i=1}^{p} A(i) \cdot b(i) - \sum_{i=1}^{p} \Omega_\mu(i)\{b(i) - b_g(i)\} - b_n(p) \sum_{i=p-\xi+1}^{p-1} A(i) \Bigg\} \cdot \Bigg\{ 1 - \sum_{i=1}^{p} A(i) \Bigg\}^{-1},$$

where

$$b_R(\leq p + \xi - 1) = \sum_{v=1}^{p} \frac{A(v)}{2b(v)} \sum_{p=1}^{l(v)} \frac{k_v(v) + 1}{k_v(v)} \cdot b_v(v)^2 \cdot q_v(v) + \sum_{v=p+1}^{p+\xi-1} \frac{A(v)}{\lambda_\mu(v)^2 \cdot b(v)} \cdot \Bigg\{ \lambda_\mu(v) \cdot b(v) - 1 + \sum_{\nu=1}^{l(v)} \frac{q_\nu(v)}{(\lambda_\mu(v) \cdot b_\nu(v)/k_\nu(v) + 1)^{k_\nu(v)}} \Bigg\}$$

is the expected time a demand of class p has to wait until demands of lower priority, which cannot be interrupted, leave the server.

$$\Omega_\mu(i) = \lambda(i) \cdot \frac{b(i) \sum_{j=1}^{i-1} A(j) - b_n(i) \sum_{j=i-\xi+1}^{i-1} A(j)}{1 - \sum_{j=1}^{i-1} A(j)}$$

is the mean number of demands of class i waiting, but being interrupted at least once.

$$b_n(p) = \frac{1}{\lambda_\mu(p)} \Bigg\{ 1 - \sum_{\nu=1}^{l(p)} \frac{q_\nu(p)}{(\lambda_\mu(p) \cdot b_\nu(p)/k_\nu(p) + 1)^{k_\nu(p)}} \Bigg\}$$

is the remaining rest-service time of a demand of class p after its last interruption.

$$b_g(v) = \frac{1}{2 \cdot b(v)} \sum_{\nu=1}^{l(v)} \frac{k_\nu(v) + 1}{k_\nu(v)} \cdot b_\nu(v)^2 \cdot q_\nu(v)$$

is the remaining service time for a demand of class i. Additionally,

$$A(p) = \lambda(p) \cdot b(p) \text{ and}$$

$$\lambda_\mu(p) = \sum_{i=1}^{p-\xi} \lambda(p).$$

Remark It should be mentioned that an explicit solution has also been found. However, from the computational viewpoint the presented recursive solution is more practical.

In addition to the expected response time and waiting time $w(p) = r(p) - b(p)$, the following characteristic performance measures have been derived:

1. Probability that a demand of class p is interrupted at least once:

$$P_\mu(p) = 1 - \sum_{\nu=1}^{l(p)} \frac{q_\nu(p)}{(\lambda_\mu(p) \cdot b_\nu(p)/k_\nu(p) + 1)^{k_\nu(p)}}.$$

2. Mean number of interrupts per demand of priority class p:

$$\mu(p) = \lambda_\mu(p) \cdot b(p).$$

3. Probability of waiting for demands of priority class p:

$$W(p) = \sum_{i=1}^{p+\xi-1} A(i) + \Bigg\{ 1 - \sum_{i=1}^{p+\xi-1} A(i) \Bigg\} \cdot P_\mu(p).$$

Appendix 2: Response time for preemption-distance priorities with fixed interrupt level (two-dimensional representation)

The expected response time for a priority class (g, γ) is obtained following precisely the same principle outlined in Appendix 1.

For brevity set $r_{g,\gamma}[\xi_1, \xi_2, \cdots, \xi_G] = r(g, \gamma)$. Then the expected response time is given by the following recursive solution:

$$r(g,\gamma) = \Bigg\{\sum_{i=1}^{g}\sum_{j=1}^{R(i,\gamma-1)} A(i,j)r(i,j) + b(g,\gamma) + b_R(\leq g, \xi_g) - \sum_{i=1}^{g}\sum_{j=1}^{R(i,\gamma)} A(i,j)\cdot b(i,j) - b_n(g,\gamma)\times\sum_{j=1}^{\gamma} A(g,j) - \sum_{i=1}^{g}\sum_{j=1}^{R(i,\gamma)} \Omega_\mu(i,j)[b(i,j) - b_g(i,j)]\Bigg\} \times\left\{1 - \sum_{i=1}^{g}\sum_{j=1}^{R(i,\gamma)} A(i,j)\right\}^{-1},$$

where

$$R(a,b) = \begin{cases}\xi_a \text{ when } a < g,\\ b \text{ when } a = g,\end{cases}$$

$$b_R(\leq g,\gamma) = \sum_{i=1}^{g}\sum_{j=1}^{R(i,\gamma)} \frac{A(i,j)}{2b(i,j)}\sum_{\nu=1}^{l(i,j)}\frac{k_\nu(i,j)+1}{k_\nu(i,j)}\cdot b_\nu(i,j)^2\cdot q_\nu(i,j) + \sum_{j=\gamma+1}^{\xi_g}\frac{A(g,j)}{\lambda_\mu(g,j)^2\cdot b(g,j)}\left\{\lambda_\mu(g,j)\cdot b(g,j) - 1 + \sum_{\nu=1}^{l(g,j)}\frac{q_\nu(g,j)}{\left[\frac{\lambda_\mu(g,j)\cdot b_\nu(g,j)}{k_\nu(g,j)}+1\right]^{k_\nu(g,j)}}\right\},$$

$$\Omega_\mu(i,j) = \lambda(i,j)\cdot\frac{b(i,j)\cdot A_\mu(i,j) + \{b(i,j) - b_n(i,j)\}\sum_{\nu=1}^{j-1}\dot{A}(i,\nu)}{1 - A_\mu(i,j) - \sum_{\nu=1}^{j-1}\dot{A}(i,\nu)},$$

$$b_n(g,\gamma) = \frac{1}{\lambda_\mu(g,\gamma)}\left\{1 - \sum_{\nu=1}^{l(g,\gamma)}\frac{q_\nu(g,\gamma)}{\left[\frac{\lambda_\mu(g,\gamma)\cdot b_\nu(g,\gamma)}{k_\nu(g,\gamma)}+1\right]^{k_\nu(g,\gamma)}}\right\},$$

$$b_g(i,j) = \frac{1}{2\cdot b(i,j)}\sum_{\nu=1}^{l(i,j)}\frac{k_\nu(i,j)+1}{k_\nu(i,j)}\cdot b_\nu(i,j)^2\cdot q_\nu(i,j), \text{ and}$$

$$\lambda_\mu(g,\gamma) = \sum_{i=1}^{g-1}\sum_{j=1}^{\xi_{g-1}}\lambda(i,j).$$

Acknowledgments

The author thanks Professor Dr.-Ing. A. Lotze, Director of the Institute for Switching Techniques and Data Processing, University of Stuttgart, for continuously supporting all investigations concerning the uniform description and analysis of preemption-distance priorities. The author is also grateful to two IBM colleagues: L. S. Woo, for many stimulating discussions on optimization, for valuable comments, and for helpful suggestions; and D. T. Tang, for his support and discussion of the manuscript.

References

1. J. Martin, *Design of Real-Time Computers*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1967.
2. M. Graef, R. Greiller and G. Hecht, *Datenverarbeitung im Realzeitbetrieb*, R. Oldenbourg Verlag, Munich, Germany 1970.
3. U. Herzog, "Preemption-Distance Priorities in Real-Time Computer Systems," *Nachrichtentechn. Z.* **25,** 201 (1972).
4. U. Herzog, "Efficient Priority Strategies for Switching Centers in Communication Networks," *Proc. Second Texas Conference on Computing Systems*, University of Texas at Austin, November 1973, p. 45.1.
5. W. Chang, "Queuing with Non-Preemptive and Preemptive Resume Priorities," *Oper. Res.* **13,** 1021 (1965).
6. I. M. Dukhovnyi and K. K. Kolin, "Optimization of the Structure of a Digital Computer Program as a Priority Queuing System," *Proc. Symp. on Computer-Communication Networks and Teletraffic*, Polytechnic Institute of Brooklyn, New York, 1972, p. 145.
7. H. M. Wagner, *Principles of Management Science*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1970.
8. H. Greenberg, *Integer Programming*, Academic Press, Inc., New York and London 1971.
9. F. Lootsma, *Numerical Methods for Non-Linear Optimization*, Academic Press, Inc., New York and London 1972.
10. D. G. Kendall, "Stochastic Processes Occurring in the Theory of Queues and Their Analysis by the Method of the Imbedded Markov Chain," *Ann. Math. Statist.* **24,** 338 (1953).
11. E. Brockmeyer, H. L. Halstrøm and A. Jensen, "The Life and Works of A. K. Erlang," *Acta Polytechnica Scandinavia* **287** (1960).
12. D. V. Lindley, "The Theory of Queues with a Single Server," *Proc. Cambridge Phil. Soc.* **48,** Part 2, 277 (1952).
13. D. R. Cox and H. D. Miller, *The Theory of Stochastic Processes*, John Wiley and Sons, Inc., New York 1965.
14. U. Herzog, "Verkehrsfluss in Datennetzen," Habilitation Thesis, University of Stuttgart, Germany, December 1973.

Received March 10, 1974

The author is located at the Institute for Switching Techniques and Data Processing at the University of Stuttgart, West Germany.