PRIORITY MODELS FOR COMMUNICATION PROCESSORS

INCLUDING SYSTEM OVERHEAD

Ulrich Herzog

Institute of Switching and Data Techniques
University of Stuttgart

Institute for Mathematical Machines and Data Processing
University of Erlangen-Nürnberg (Since May 76)

Fed. Rep. Germany

ABSTRACT

Various scheduling strategies for communication processors are discussed and uniformly described by means of the so-called "Preemption Distance".
Then, two types of communication processors are described: processors with fully automatic (hardware) priority-interrupt systems and, secondly, processors with software controlled priority-interrupt systems. For both, queuing models including Preemption-Distance Priorities are presented and analyzed.
Numerical results show how software overhead influences the characteristic performance values.
The paper contains the following sections:
1. Introduction
2. Classification of Preemption-Distance Priority Strategies
3. Modelling of Communication Processors
4. Analysis
5. Numerical Results
6. Summary and Outlook

1. INTRODUCTION

In order to fulfill response time constraints it is necessary to handle special pressing demands (alarms, routing information, urgent messages, etc.) by means of preemptive priorities. On the other hand, there are less urgent demands which do not justify interruptions. 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 a mixed mode. Typical examples are
- the electronic telephone switching system EWS 1 /1/,
- the electronic data switching centre EDS /2/,
- controllers of IBM teleprocessing systems /3/,
- many real-time computers, such as the PDP-11 family /4,5/, AEG-60 family /6/, etc.

Most theoretical investigations neglect this fact and deal either with pure preemptive or pure nonpreemptive (head-of-the-line) priorities.
The author demonstrated in /7,8/ how arbitrary combinations of preemptive and nonpreemptive priorities can be uniformly described by means of the so-called "Preemption Distance"; waiting demands of different classes are served in the order of their priority ("STRICT" loading procedure).
For small and medium size processors it is often economic to process interrupted programs before the newly arrived programs even if these programs are of higher (nonpreemptive) priority ("IPF" loading procedure).
These strategies as well as all special cases known from literature/3,9,10/ are now included in the general concept of Preemption-Distance priorities.
Queuing models are presented for both, processors with fully automatic (hardware) priority-interrupt systems and processors with software controlled priority-interrupt systems.

2. CLASSIFICATION OF PREEMPTION-DISTANCE PRIORITIES

2.1. GENERAL REMARKS

Scheduling strategies for real-time computers are characterized by two attributes:
- the interrupt procedure;
  it describes whether a newly arriving demand is allowed to interrupt the current program, and
- the loading procedure;
  it determines the next (waiting) demand to be processed after completion of a program.

The author demonstrated in /7,8/ how to describe uniformly a wide class of interrupt procedures by means of the so-called "Preemption-Distance". A short summary is given in section 2.2.
Several loading procedures are described in section 2.3.

2.2. INTERRUPT PROCEDURES

2.2.1. Two Examples

The most important requirement of real-time data processing is the ability of the system to react quickly to paricularly urgent demands. This is possible with the use of pure preemptive priorities.
Each interruption, however, requires some additional system overhead, e.g.
- saving of registers
- identification, analysis and processing of interrupt requests,
- reorganisation of queues, etc.

In order to keep this additional system load as small as possible, modern real-time data processing systems use reasonable combinations of preemptive and nonpreemptive (head-of-the-line) priorities.
Most often implemented are interrupt procedures with fixed interrupt levels: All demands are classified into some few priority groups ("interrupt levels") which interrupt each other (preemptive priorities). Within one interrupt level demands may occur with different priorities. However, they do not interrupt each other /1, 3-6/.
Figures 1, 2 show a small but typical example. Dependent on the importance of a demand, the distance varies to the next class of priorities to be interrupted - the so-called Preemption-Distance.
A more sophisticated strategy is shown in figure 3: demands of class 15 interrupt demands of classes 18, 19. etc., however not the intermediate classes 16 and 17 whereas demands of class 16 interrupt only classes 21, 22, etc./2/.

Eighth Intenational Teletraffic Congress (ITC)
Melbourne, Australia, November 1976

WEITERVERWENDUNG NUR
MIT QUELLENANGABE!

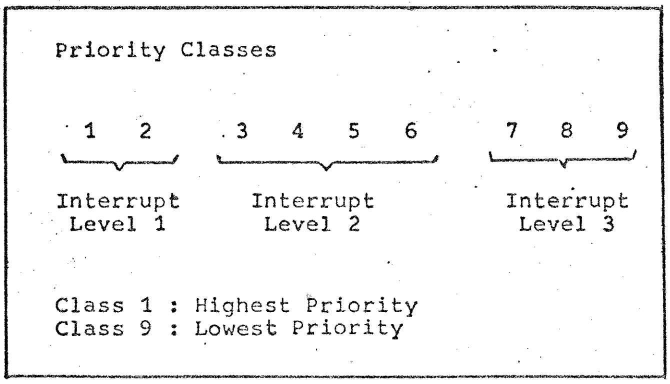


Fig.1 : First example for a combination of both preemptive and nonpreemptive priorities.

Newly arrived demand of class	does not interrupt service of class	interrupts service of class	Preemption Distance
1	1,2	3,4,5,6,7,8,9	2
2	1,2	3,4,5,6,7,8,9.	1
3	1,2,3,4,5,6	7,8,9	4
4	1,2,3,4,5,6	7,8,9	3
5	1,2,3,4,5,6	7,8,9	2
6	1,2,3,4,5,6	7,8,9	1
7	1,2,3,4,5,6,7,8,9	-	-
8	1,2,3,4,5,6,7,8,9	-	-
9	1,2,3,4,5,6,7,8,9	-	-

Fig.2 : Service mechanisme and preemption distance corresponding to the priority strategy of fig.1 .

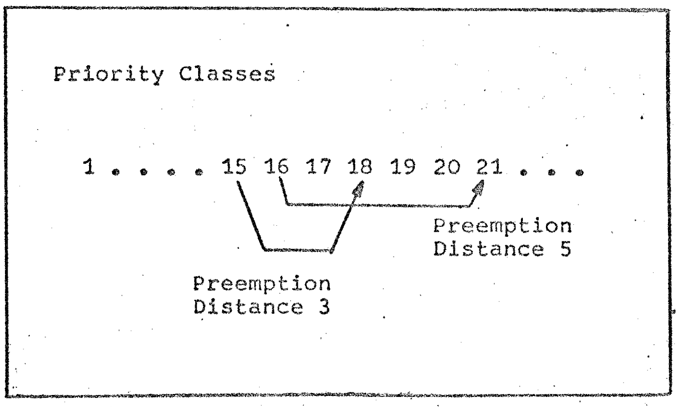


Fig.3 : Second example for a combination of preemptive and nonpreemptive priority strategies.

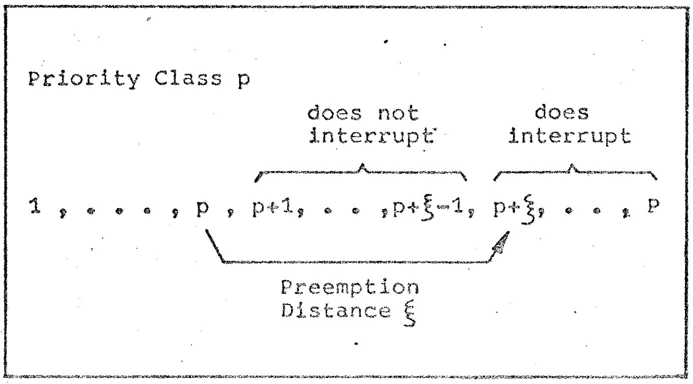


Fig.4 : Introduction of the Preemption Distance. The special cases, preemptive priorities (ξ= 1) and nonpreemptive priorities (ξ= P), are included.

2.2.2. Description of Preemption-Distance Priorities

2.2.2.1. Uniform Preemption-Distance for all Classes, Preemptive Priorities, Nonpreemptive (Head-of-the-line) Priorities.

A general class of priority strategies can be characterized by the Preemption Distanceξ, the uniform distance between a priority class and the next priority class being interrupted. Figure 4 illustrates this definition: Demands of class p(p= 1,2,...,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).
It is easily seen that two well known special cases of Preemption-Distance Priorities are included:
ξ= 1: preemptive priorities
ξ= P: nonpreemptive (head-of-the-line) priorities

2.2.2.2. Arbitrary, Nonuniform Preemption Distance for each Priority Class

The Preemption Distance ξ(p) may be defined individually for each priority class p(p=1,2,..., P). Then arbitrary combinations of preemptive and nonpreemptive 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. For more details, cf. /7,8/.

2.2.2.3. Fixed Interrupt Levels

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 /1, 3-6/. These interrupt procedures may be described according to section 2.2.2.2. Sometimes, however, a twodimensional notation is more advantageous. A detailed description and several examples are also presented in /8/.

2.3. LOADING PROCEDURES

The loading procedure determines, after completion of a program, the next program to be processed. Two strategies are distinguished
. STRICT, the classical loading procedure,
. IPF, a modified loading procedure.

2.3.1. "STRICT", the Classical Loading Procedure

After completion of a program, waiting demands of different classes are strictly served in the order of their priority (FIFO within one class). Fig. 5 shows a typical example for the sequence of programs being processed.
It is obvious, that this loading procedure guarantees the fastest reaction to urgent requests.
The disadvantage is, however, that a maximum number I of programs may be interrupted at the same time ($I=P-\xi$ for uniform preemption distance).

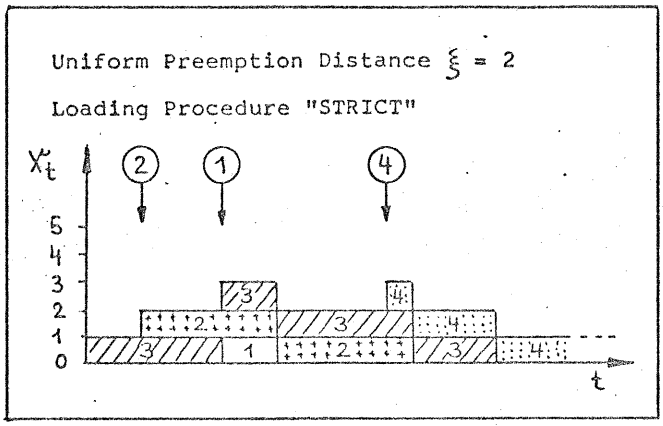


Fig.5 : Example for the influence of the loading procedure "STRICT" on job-scheduling (X_t :number of jobs in the system at time t; one processor is assumed).

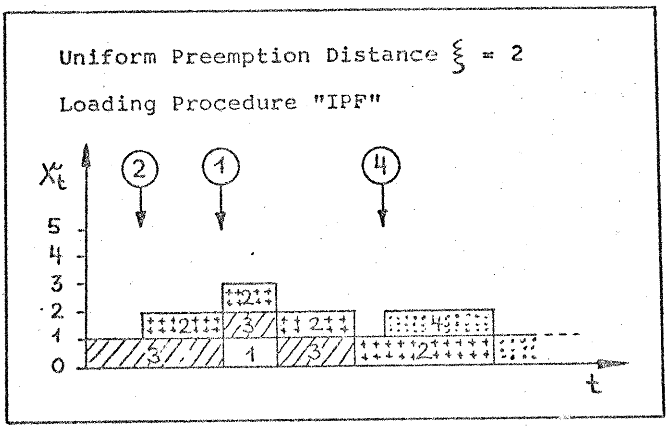


Fig.6 : Example for the influence of the loading procedure "IPF" on job-scheduling (X_t :number of jobs in the system at time t waiting or being processed; one processor is assumed).

2.3.2."IPF", Interrupted Programs First

Interrupted programs, if any, are serviced before the newly arrived programs even if these programs are of higher (nonpreemptive) priority. Fig. 6 shows the sequence of processing according to this strategy.
The system-response is slightly slower to urgent requests (cf. also section 5). The maximum number of interrupted programs is reduced to

$$I = \left[\frac{P-1}{\xi} \right]$$

and so the number of parallel register sets, storage place for interrupt stacks, etc. (uniform preemption distance). Therefore, this loading procedure is most economic for many small and medium size systems.

3. MODELLING COMMUNICATION PROCESSORS

3.1. STRUCTURE AND OPERATING MODE

Today there exists a wide variety of real-time processors between the following extremes /5/

- processors with fully automatic (hardware) priority-interrupt systems, and
- processors with software controlled priority-interrupt systems.

the main features of both types are described and queuing models derived.
Processors between these two extremes may be modelled and analyzed, accordingly. The results, presented above, however, show clearly the lower and upper bounds for system performance.

3.1.1. Hardware Interrupt Systems.

The main features of a fully automatic priority-interrupt system are:

- an interrupt module which provides for arming/disarming individual interrupts and enabling/disabling recognition of interrupts (masking),
- multiple register sets
- individual interrupts each with its own unique memory address
- automatic nesting of routines and reentrant coding

So system overhead for interrupts is small compared to processing times. Therefore, processors with fully automatic priority-interrupt systems can be modelled by means of

- single-server queuing systems with parallel input queues and Preemption-Distance Priorities of arbitrary type, cf. fig. 7.

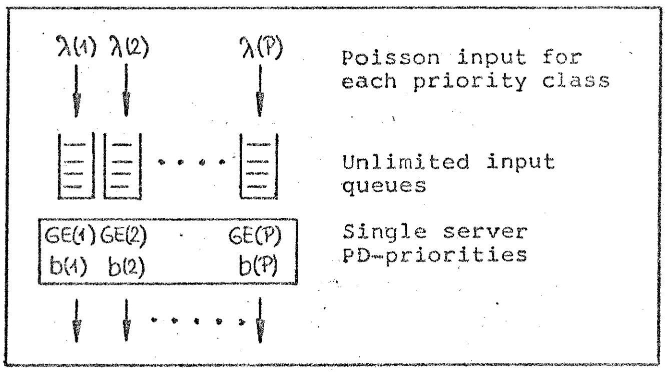


Fig.7 : Queuing model for processors with a fully automatic (hardware) interrupt system (λ: arrival rates; b: mean service times; GE: General Erlangian distribution).

3.1.2. Software Controlled Interrupt Systems

In the most primitive interrupt system simply all interrupt requests are put onto a single line. Hardware gives the processor control to the software system.
Contents of the program counter and registers are saved in specific memory locations. Then, the interrupt request is identified, analyzed and, if necessary, suppressed. If there is no suppression processing of the interrupt request is initiated.
The routine ends by restarting program counter and registers to return to the original program.
Figure 8 surveys the various software and hardware operations; for more details cf. /5/.

The appropriate queuing model is shown in figure 9: Arriving demands of class i (i=1,2,...,P) are stored in the input buffers. At the same time an interrupt request is generated. Interrupt requests (class o) interrupt all other classes. After interrupt processing the newly arrived demand is put into the corresponding

input queue. The scheduling strategy for the input queues are Preemption-Distance Priorities of arbitrary type.
Remark: Again, it is easy to modify this queuing model, to distinguish several types of interrupt requests, etc.

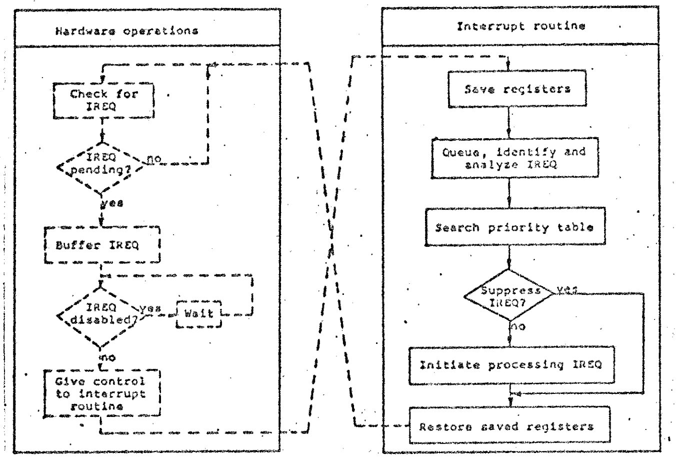


Fig.8 : Example for the various operations of a software controlled interrupt system (IREQ: interrupt request).

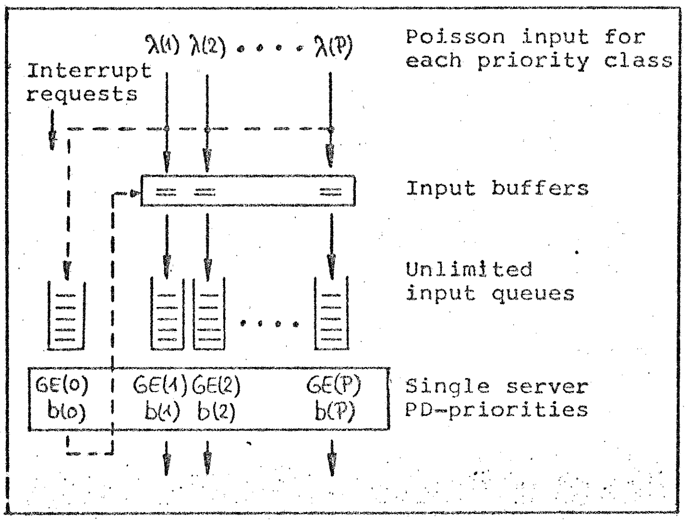


Fig.9 : Queuing model for processors with a software controlled interrupt system.

3.2. TRAFFIC PARAMETERS

Arrival processes are assumed to be Poissonian. Service times , individually for each priority class and overhead phases are distributed according to a "General Erlangian" distribution (GE). Therefore, any distribution of actual service and overhead times can be modelled with any required accuracy.

Arrival process:

$$F_p^A(t) = P(T_{A_p} \leq t) = 1 - e^{-\lambda(p)\cdot t}$$

Service- and overhead time distributions:

$$F_p^B(t) = P(T_{B_p} \leq t) \qquad p = 0, 1, \ldots, P$$

$$F_p^B(t) = \sum_{\nu=1}^{\ell(p)} q_\nu(p)\left(1 - e^{-\frac{t}{b_\nu(p)/k_\nu(p)}} \sum_{\eta=0}^{k_\nu - 1} \frac{\left(\frac{t}{b_\nu(p)/k_\nu(p)}\right)^\eta}{\eta!}\right.$$

Main abbreviations:
$\lambda(p)$: arrival rate for priority class p, p= 1,...P
$\lambda(o)$: arrival rate for overhead phases

$$\lambda(o) = \sum_{p=1}^{P} \lambda(p)$$

b(p) : mean service time for priority class p
b(o) : mean overhead time
A(p) : offered traffic A(p) = λ(p).b(p)
More abbreviations, cf. text and /8/.

4. ANALYSIS

4.1. GENERAL REMARKS

The most famous methods to investigate the stochastic behavior of such non-Markovian queuing systems are the method of embedded Markov chains /11/, the phase method /12/, the integral method /13/, and the ~~method of supple~~mentary variables /14/, cf. also /15-18/.

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 solutions can be obtained.

4.2. HARDWARE INTERRUPT SYSTEMS

4.2.1. Preemption-Distance Priorities (STRICT)

These classes of Preemption-Distance priorities have been analyzed earlier, cf. /7,8/.

4.2.2. Preemption-Distance Priorities (IPF)

The response time (time spent in the system, waiting and being processed) for a demand of priority class p(p=1,2,..., P) may be devided into two periods /16/:
- the initial waiting time, and
- the completion time.

From a didactic point of view it is advantageous to analyze at first the completion time.

4.2.2.1. The Completion Time

The completion time $T_c(p)$ for an individual demand of priority class p is the period between the first moment of service and the moment when it leaves the system.
Obviously:

$$T_c(p) = T_B(p) + \sum_{i=1}^{p-\xi} \sum_{j=1}^{N_i} T_{B_j}(i)$$

where N_i is the number of demands arriving during the completion time and $T_{Bj}(i)$ is the service time of customer no j. Introducing expectation values, the mean completion time is given by

$$t_c(p) = E\left[T_c(p)\right]$$

$$t_c(p) = b(p) + \sum_{i=1}^{p-\xi} \lambda(i) \cdot t_c(p) \cdot b(i)$$

and therefore

$$t_c(p) = \frac{b(p)}{1 - \sum_{i=1}^{p-\xi} A(i)}$$

The mean number of demands waiting and being processed at least once is, according to Little's theorem:

$$\Omega_u(p) = \lambda(p) \cdot \left(t_c(p) - b(p)\right)$$
$$= A(p) \cdot \frac{\sum_{i=1}^{p-\xi} A(i)}{1 - \sum_{i=1}^{p-\xi} A(i)}$$

4.2.2.2. The Initial Waiting Time

The initial waiting time is the waiting time before the first period of service. Following the same arguments as above, the mean initial waiting time $t_I(p)$ is composed of the following four terms:

- remaining service time: at the time of arrival, a demand of class i is being processed with probability $A(i) = \lambda(i) \cdot b(i)$. It is interrupted immediately, if i≥p+§. Otherwise service is completed before the newly arrived demand gets its first period of service:

$$t_{W1}(p) = \sum_{i=1}^{p+\xi-1} A(i) \cdot b_g(i) = \sum_{i=1}^{p+\xi-1} A(i) \cdot \frac{b(i)}{2}\left(1 + \frac{d^2(i)}{b(i)^2}\right)$$
$$= \sum_{i=1}^{p+\xi-1} A(i) \cdot \frac{M_2(i)}{2b(i)}$$

where $\sigma^2(i)$ is the variance and $M_2(i)$ the second moment of the service time distribution for class i.

Waiting time $t_{w2}(p)$: all waiting demands of higher or equal priority are serviced before the newly arrived demand. Some of them are partially serviced already (remaining service time $b_g(i)$) the rest waits for its first service (total service time $b(i)$):

$$t_{W2}(p) = \sum_{i=1}^{p} \Omega_u(i) \cdot \frac{M_2(i)}{2b(i)} + \sum_{i=1}^{p} \left(\Omega(i) - \Omega_u(i)\right) \cdot b(i)$$

where

$$\Omega(i) = \lambda(i) \cdot (\tau(p) - b(p))$$
$$= \lambda(i) \cdot (t_i(p) + t_c(p) - b(p))$$

according to Little's theorem.

Waiting time $t_{w3}(p)$: according to the IPF-strategy demands of nonpreemptive priority i(p+1≤i≤p+§-1) waiting and being interrupted at least once are serviced before the newly arrived demand:

$$t_{W3}(p) = \sum_{i=p+1}^{p+\xi-1} \Omega_u(i) \cdot \frac{M_2(i)}{2b(i)}$$

- Waiting time $t_{w4}(p)$: all demands of higher priority (i≤p-1) arriving during the initial waiting time are serviced before the newly arrived demand:

$$t_{W4}(p) = \sum_{i=1}^{p-1} \lambda(i) \cdot t_i(p) \cdot b(i)$$

Summing up all terms a recursion formula for the mean initial waiting time $t_I(p)$ is obtained, whose solution is:

$$t_I(p) = \sum_{i=1}^{p+\xi-1} \frac{A(i) \cdot M_2(i)}{2b(i)\left(1 - \sum_{j=1}^{p} A(j)\right)\left(1 - \sum_{j=1}^{p-1} A(j)\right)}$$

Remark: The same result may be obtained directly using a theorem on the busy period of systems (Adiri /19/, Kleinrock /20/). Properly applied it shows that the length of the initial waiting time is not affected by the scheduling strategy. The above equation is known for nonpreemptive priorities with (p+§-1) classes!

4.2.2.3. The Mean Response Time

The response time is determined by the sum of initial waiting time and completion time. Hence, the mean response time for class p is:

$$\tau(p) = \sum_{i=1}^{p+\xi-1} \frac{A(i) \cdot M_2(i)}{2b(i)\left(1 - \sum_{j=1}^{p} A(j)\right)\left(1 - \sum_{j=1}^{p-1} A(j)\right)} + \frac{b(p)}{1 - \sum_{j=1}^{p-\xi} A(j)}$$

Remark: It is easily shown that the results of Gay and Seaman /3/ obtained by heuristic reasoning for systems with fixed interrupt levels are included in the above exact solution /21/.

4.3. SOFTWARE CONTROLLED INTERRUPT SYSTEMS

4.3.1. Preemption-Distance Priorities (STRICT)

Each newly arriving demand generates an interrupt request. Interrupt requests interrupt all other programs (cf. section 3). After interrupt processing the programs are strictly serviced in the order of their priority. This reordering distroyes the advantage of preemption-distance priorities and the strategy corresponds to pure preemptive priorities with system overhead, a result obtained in 4.3.2. as a special case.

Remark: A modified version allows reordering only if a running program is actually interrupted by an other program of more importance /21/.

4.3.2. Preemption-Distance Priorities (IPF)

Three periods determine the fate of a newly arrived demand

- the buffer time in the input buffer,
- the initial waiting time, and
- the completion time.

These portions of the total response time are obtained following precisely the same principle outlined in section 4.2. However, it has to be taken into account that both the actual demand and the interrupt request are generated at the same moment (group arrival). Then, the following formula is obtained for the mean response time:

$$d(p) = \frac{b(0)\left(1 - A(0)\right) + \sum_{i=0}^{p+\xi-1} \frac{A(i)\cdot M_2(i)}{2b(i)}}{\left(1 - \sum_{j=0}^{p} A(j)\right)\cdot\left(1 - \sum_{j=0}^{p-1} A(j)\right)} + \frac{b(p)}{1 - \sum_{j=0}^{p-\xi} A(j)}$$

5. NUMERICAL RESULTS

5.1. COMPARISON BETWEEN "STRICT" AND "IPF"

Many numerical examples have been studied in order to compare the influence of the loading procedures. Figures 10 and 11 show a typical example: usually there is only a small difference between the response times. STRICT favours a fast response time for higher priority demands within each interrupt level whereas IPF smoothes these differences.

5.2. INFLUENCE OF SYSTEM OVERHEAD

Figure 12 shows the influence of increasing overhead (increasing software time per interrupt handling) on the mean response time.

6. SUMMARY AND CONCLUSIONS

The advantage of Preemption-Distance Priorities is obvious. They guarantee fast reaction to urgent signals avoiding large overhead. Therefore, preemption-distance priorities are implemented in many real-time systems.

Distinguishing between several loading procedures, new strategies have been included in the general description and analysis.

Modelling two types of processors, an accurate estimation of system performance is possible.

It has been pointed out already that processors with a specific interrupt structure and interrupt software may be modelled and analyzed accordingly.

Probability of waiting, probability of interruption, system utilization, total overhead, etc. can be easily derived /21/. Although more tedious, higher moments of the waiting - and response time distribution may also be obtained.

ACKNOWLEDGMENTS

The author wishes to express his thanks to his collegue W. Bux and Mr. H. Schneider for many stimulating discussions. Preparing a study thesis, Mr. Schneider analyzed very carefully the different models, found explicit solutions and performed all numerical evaluations.

REFERENCES

/1/ Siemens: System IV, ein Fernsprechvermittlungssystem mit gespeichertem Steuerprogramm.
Reprints from "Informationen Fernsprechvermittlungstechnik", Siemens AG, Munich, 1970.

/2/ Gabler, H.G.: The German EDS Network.
Proc. ACM/IEEE Second Symp. on Problems in the Optimization of Data Commun. Systems.
Palo Alto, Calif. Oct. 1971, pp. 80-85.

/3/ Gay, T.W., Seaman, P.H.: Composite Priority Queue. IBM J. Res. Develop., Vol.19 (1975) pp. 78-81.

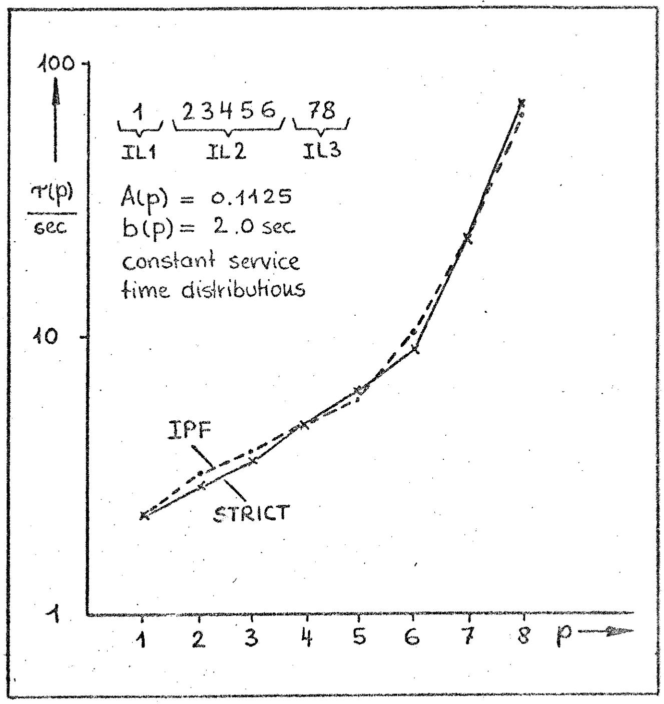


Fig.10 : Mean response time for Preemption Distance priorities with fixed interrupt levels and the two loading procedures STRICT and IPF.

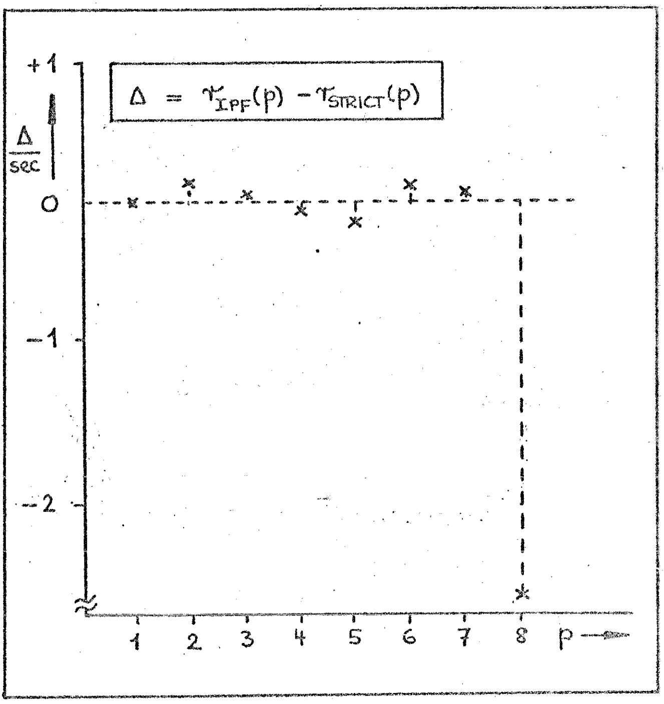


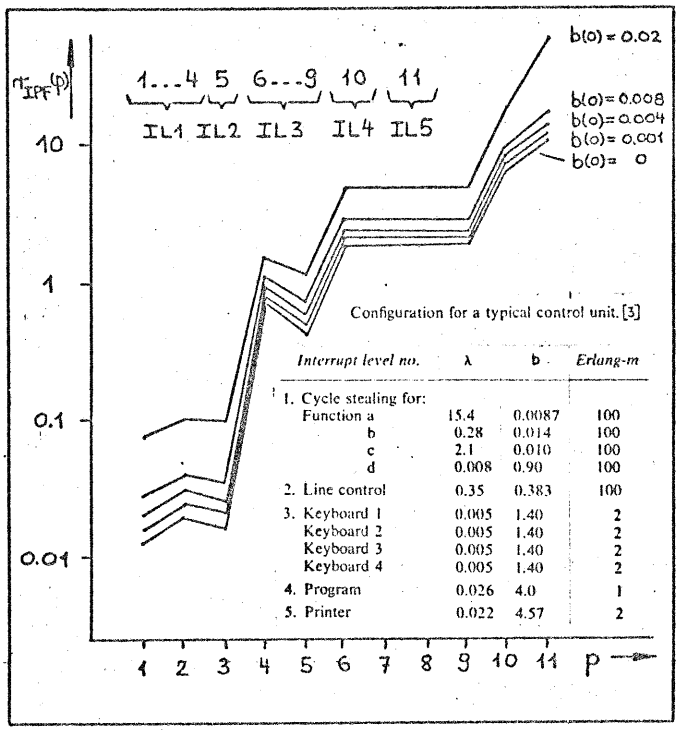
Fig.11 : Absolute difference between the two loading procedures. Traffic parameters, cf. figure 10.

/4/ Digital Equipment Corp.: PDP-11, processor-peripherals-, and interface handbooks.
Digital Equipment Corporation, 1972.

/5/ Korn, G.A.: Minicomputers for Engineers and Scientists. McGraw-Hill Book Comp., New York 1973

/6/ AEG-Telefunken: Prozessrechner AEG-60, Systemkurzbeschreibungen. AEG-Telefunken, Seligenstadt, FRG.

/7/ Herzog, U.: Preemption-Distance Priorities in Real-Time Computer Systems.
Nachrichtentechn. Z. 25(1972) pp. 201-203.



Configuration for a typical control unit. [3]

Interrupt level no.	λ	b	*Erlang-m*
1. Cycle stealing for:			
Function a	15.4	0.0087	100
b	0.28	0.014	100
c	2.1	0.010	100
d	0.008	0.90	100
2. Line control	0.35	0.383	100
3. Keyboard 1	0.005	1.40	2
Keyboard 2	0.005	1.40	2
Keyboard 3	0.005	1.40	2
Keyboard 4	0.005	1.40	2
4. Program	0.026	4.0	1
5. Printer	0.022	4.57	2

Fig.12 : The influence of software overhead on the mean response time for Preemption Distance priorities with IPF loading procedure ("Strict" cf. section 4.3.1.). Numerical values changed insignificantly when different distribution functions were chosen for the overhead phases.

/8/ Herzog, U.: Optimal Scheduling Strategies for Real-Time Computers. IBM Thomas J. Watson Research Centre, Research Report RC 4889, Yorktown Heights, N.Y.,1974 and IBM J. Res. Develop., Vol 19, September 1975.

/9/ Chang, W.: Queuing with Non-Preemptive and Preemptive Resume Priorities. Operations Res. 13 (1965) pp. 1021-1022.

/1o/Dukhovnyi, I.M., Kolin, K.K.: Optimization of the Structure of a Digital Computer Program as a Priority Queuing System. Proc. Symp. on Computer-Commun. Networks and Teletraffic, Polytechn. Inst. of Brooklyn, New York (1972), pp. 145-156.

/11/Kendall, D.G.: Stochastic Processes Occuring in the Theory of Queues and their Analysis by the Method of the Imbedded Markow Chain. Ann.math. Statist 24(1953), pp. 338-354.

/12/Brockmeyer, E., Halstrøm, H.L., Jensen, A.: The life and works of A. K. Erlang. Acta Polytechnica Scandinavia (1960), Nr. 287.

/13/Lindley, D.V.: The Theory of Queues with a Single Server. Proc. Cambridge Phil.Soc. Vol.48, part 2, 1952.

/14/Cox, D.R., Miller, H.D.: The Theory of Stochastic Processes. J.Wiley & Sons Inc., New York, 1965.

/15/Cobham, A.: Priority Assignment in Waiting Line Problems. Operations Res. 2 (1954) pp. 70-76 and 3 (1955) pp. 547.

/16/Gaver, D.P.: On Priority Type Disciplines in Queuing . Proc. Symp. on Congestion Theory (1964). The University of North Carolina Press, Chapel Hill, 1965.

/17/Takács, L.: Priority Queues. Operations Res. 12 (1964) pp. 64-74.

/18/Jaiswal, N.K.: Priority Queues. Academic Press, New York and London, 1968.

/19/Adiri, I.: Introduction to Queuing Theory with Applications to Computer Systems. Computer Science Dept. Monograph Series, RA 42, IBM Research Division, Yorktown Heights, New York, 1972.

/20/Kleinrock, L.: Queuing Systems. Vol.1: Theory. John Wiley and Sons, New York, 1975 417 pp.

/21/Schneider, H.: Verallgemeinerung der Unterbrechungs-Distanz Prioritäten. Study thesis, University of Stuttgart, 1976.