ITC 11, Minoru Akiyama (Editor)
Elsevier Science Publishers B. V. (North-Holland)


# EFFICIENCY OF SCHEDULING ALGORITHMS FOR SWITCHING SYSTEMS WITH DISTRIBUTED CONTROL

Wolfgang DENZEL

Institute of Communications Switching and Data Technics
University of Stuttgart, Fed. Rep. of Germany

ABSTRACT

The paper deals with problems on the intermodule communication in switching systems with distributed and modularized control. Two basic scheduling aspects are regarded in the path between an output queue of a processor and the input queue of another processor, namely intelligent polling mechanisms for output queues as well as intelligent task dispatching strategies for load sharing processors. In both cases, performance results are obtained by analytic means for generic queueing models of state dependent dynamic strategies. Comparisons between different simple and intelligent strategies referring to, e.g., the blocking probability, mean waiting time, control overhead, and implementation aspects allow the choice of the suitable strategy and the dimensioning of its parameters under given conditions.

## 1. INTRODUCTION

The control of digital switching systems becomes more and more distributed by using a large number of microprocessors. Flexibility, extensibility and availability can be obtained by this modularization in the control area. On the other hand, the problems of intermodule communication and the distribution of functions become more difficult.

A generalized model of such a control structure (see Fig. 1) consists of several functional groups of processors, e.g. special peripheral processors, signalling processors or universal processors for more centralized functions. By this, the principle of function sharing is implemented, but within such functional groups of processors the principle of load sharing can be used to reach flexibility and extensibility. The interprocessor communication usually is realized by sending messages into an output queue of the origin processor, from where they are transferred to the input queue of the destination processor by a communication system.

Assuming a centralized communication system, e.g. a bus, an effective access protocol is necessary. An often implemented method is some kind of polling. In the model this is represented by the lower switch. In consequence of the principle of function sharing, but also in the case of momentary overload, intelligent scheduling strategies for the outgoing messages can improve the performance, e.g. a dynamic state dependent polling mechanism. Such strategies are shortly considered in chapter three.

On the other side, the principle of load sharing implies a second scheduling problem, namely an intelligent message dispatching mechanism, e.g. a dynamic state dependent load balancing strategy. This is represented in the model by the upper switch. Different strategies are investigated in detail in chapter two.

Both mentioned intelligent scheduling strategies can commonly be implemented, e.g., in a centralized communication system control, as the bus control in the regarded model. In [1] an integrated switching system with a control structure according to Fig. 1 has been presented. The performance of the whole system has been investigated there by simulation. Since a detailed simulation technique of the whole system is not the suitable method to investigate many different scheduling strategies, this is done in this paper by analytic means for generic models of the two considered scheduling problems. The results allow the choice of the optimal strategy and the dimensioning of its parameters for both cases. So, the whole system can be investigated with the chosen strategy either by simulation or probably by analytic approximations, which eventually can be applied, because simplifications are allowed, if optimal scheduling strategies are used. This could be, e.g., the use of an equivalent single queue model as a replacement for a multiqueue submodel.

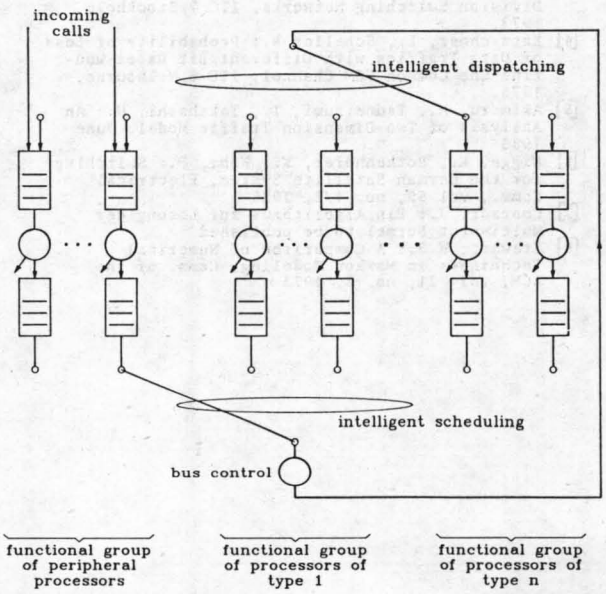


Fig. 1 Generalized queueing model of a modularized switching control

## 2. PERFORMANCE OF DISPATCHING STRATEGIES

### 2.1 Dispatching Strategies

Dispatching strategies are taken into consideration, if there exist several processors for a set of functions according to the principle of load sharing. Simple dispatching strategies are regarded here for comparisons with intelligent strategies. In the last case, the dispatcher uses system state informations for the decision, where a message has to be assigned to. Service time oriented strategies are neglected here, as in general the service times cannot be derived from the type of message. Dispatching priorities can also be neglected, because load sharing processors usually are of the same type.

The following dispatching mechanisms are considered:

Strategy 1: Random dispatching, according to assignment probabilities (e.g. equally distributed).

Strategy 2: Ordinary cyclic dispatching.

Strategy 3: Fully dynamic (state dependent) dispatching, that means "join the shortest queue" strategy. In case of equality of shortest queue lengths the dispatching follows, e.g.,
- a) strategy 1, equally distributed, or
- b) strategy 1, but deterministically to, e.g., the first of the shortest queues, or
- c) strategy 2, cyclically, e.g. relative to the last assignment.

Strategy 4: Partially dynamic (state dependent) dispatching, that means according to, e.g., strategy 1, equally distributed among those queues, whose queue length is below a given threshold, and if all queue lengths exceed the threshold the dispatching follows, e.g.,
- a) strategy 1, equally distributed, or
- b) strategy 3a.

Strategy 5: Dispatching with second attempt, that means according to, e.g., strategy 1, equally distributed. But if the selected queue is full, one second attempt is possible, e.g., according to
- a) strategy 1, equally distributed among all other queues, or
- b) strategy 1, equally distributed among those queues which are not full, or
- c) strategy 3a.

Strategy 1 is regarded, because its analysis is very simple, but in practice it is not so easy to implement as strategy 2. The intelligent strategy 3 is the optimal strategy referring to blocking and waiting time, but its implementation is difficult and the control overhead is high, because the dispatcher has to know the whole system state at any time. Therefore, strategy 4 and strategy 5 are considered, where the dispatcher not so often needs so much of the system state as in strategy 3. The implementation is easier than for strategy 3, especially for strategy 5.

### 2.2 Basic Queueing Model

Fig. 2 shows the generic queueing model for the investigations of dispatching strategies. A message arriving with rate $\lambda$ is assigned to one of g queues according to the special dispatching strategy. It gets blocked with probability B, if it cannot be assigned to the queue or the queues, which were defined by the dispatching strategy. A given queue i has $s_i$ waiting places and the appropriate server has the serving rate $\varepsilon_i$. The arrival and service process must be of Markovian type (M), that a state space analysis for the intelligent strategies can be applied. This restriction is indeed not too bad for applications in switching controls. This has been shown in the detailed simulation study of a switching system in [1]. Measurements taken at various points in the model have shown that traffic streams can be described by coefficients of variation for the interevent time between 0.9 and 1.1 in most cases.

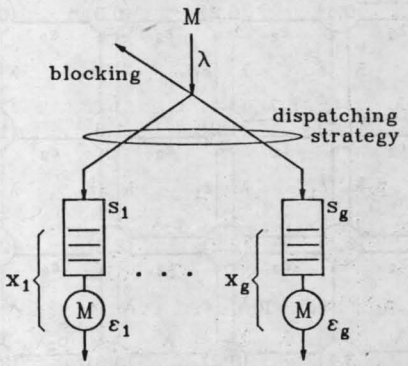


Fig. 2 Basic model for dispatching strategies

### 2.3 Solution Method

#### 2.3.1 The State Space

The solution of strategy 1 can be reduced to the solution of a M/M/1-s delay-loss system with arrival rate $\lambda_i = p_i \cdot \lambda$, where $p_i$ is the dispatching probability, e.g. $p_i = 1/g$ for equally distributed random dispatching. The characteristic values can be obtained, e.g., from tables [2].

The investigation of strategy 2 can be reduced to the solution of an $E_g/M/1$-s delay-loss system with Erlang-g arrival process and rate $\lambda_i = \lambda/g$ as above. The characteristic values can easily be obtained by a recursive solution of an appropriate two-dimensional state space.

For all other intelligent strategies state spaces have to be set up with the appropriate state transition rates. For all these cases a state $\underline{X}$ must be at least a g-dimensional vector, where $\underline{X} = (x_1, x_2, \ldots, x_i, \ldots, x_g)$; $x_i$ represents the momentary queue length of queue i, including the server. Strategies with cyclic components must have states of one more dimension for the cycle number.

Fig. 3 shows a simple example of the state space for two queues, each with three waiting places and for the dynamic strategy 3. Each state is represented by the vector $\underline{X} = (x_1, x_2)$. In case of equality of the queue lengths, messages are assigned to queue 1 with the probability $p_1$ and to queue 2 with $p_2 = 1-p_1$. For strategy 3a $p_1$ and $p_2$ have a value of 0.5 and for strategy 3b $p_1$ is 1.0 and $p_2$ is 0.

In practical applications there are normally more than two queues with some more waiting places and therefore some hundred or thousand of states. Furthermore, the multi-dimensional state spaces can no more be described in a graphical manner. Thus, a special software tool has been implemented, which allows on the one side a simplified interactive input of states and transi-

tions for smaller but complicated state spaces or, on the other side, the definition of great state spaces by an algorithmic description of the transitions. The program produces the conditional state transition probabilities for arrivals as well as directly the system of linear state equations for the statistical balance. Then the state probabilities can be solved by an iterative Gauss-Seidel algorithm. Out of these state probabilities and the state transition probabilities, characteristic mean values of the dispatching strategies can be calculated as follows.

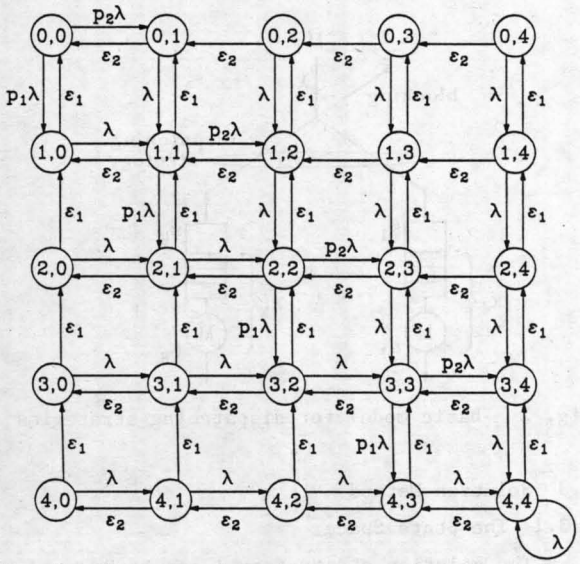


Fig. 3 Example of a state space for dispatching strategy 3

### 2.3.2 The Characteristic Values

The equations in this paragraph can be applied for state dependent dispatching strategies, as the strategies 3, 4 and 5. Thereby the following assumptions are made:

| | |
|---|---|
| $g$ | number of queues, |
| $n_i$ | number of waiting places of queue i, including the server ($n_i = s_i+1$), |
| $\underline{X}$ | state $(x_1,x_2,\ldots,x_g)$, |
| $p(\underline{X})$ | state probability of the state $\underline{X}$, |
| $p_i(\underline{X})$ | conditional state transition probability for the transition from state $\underline{X}$ to state $(x_1,x_2,\ldots,x_i+1,\ldots,x_g)$. |

The following characteristic values can be calculated for every subsystem i and some of them also as global values for the whole system:

- Probability, that an arriving customer is assigned to queue i:

$$q_i = \sum_{\underline{X}} p(\underline{X}) \cdot p_i(\underline{X}) \qquad (1)$$

- Arrival rate at queue i:

$$\lambda_i = q_i \cdot \lambda \qquad (2)$$

- Blocking probability at queue i conditioned that a call is assigned to queue i and total blocking probability, i.e. the probability that an arriving call is rejected:

$$B_i = 1/q_i \cdot \sum_{\underline{X}|x_i=n_i} p(\underline{X}) \cdot p_i(\underline{X}) \qquad (3a)$$

$$B = \sum_{i=1}^{g} q_i \cdot B_i \qquad (3b)$$

- Mean load of server i and total mean load of the whole system:

$$Y_i = 1 - \sum_{\underline{X}|x_i=0} p(\underline{X}) \qquad (4a)$$

$$Y = \sum_{i=1}^{g} Y_i \qquad (4b)$$

- Mean queue length of queue i and total mean number of waiting customers in the whole system:

$$\Omega_i = \sum_{\underline{X}|x_i>0} (x_i-1) \cdot p(\underline{X}) \qquad (5a)$$

$$\Omega = \sum_{i=1}^{g} \Omega_i \qquad (5b)$$

- Mean waiting time (with respect to all arrivals in queue i) and global mean waiting time for an arriving call at the system:

$$w_{l\,i} = \Omega_i / \lambda_i \qquad (6a)$$

$$w_l = \Omega / \lambda \qquad (6b)$$

- Waiting probability in queue i and global waiting probability for an arriving customer at the system:

$$W_i = 1/q_i \cdot \sum_{\underline{X}|0<x_i<n_i} p(\underline{X}) \cdot p_i(\underline{X}) \qquad (7a)$$

$$W = \sum_{i=1}^{g} q_i \cdot W_i \qquad (7b)$$

- Mean waiting time (with respect to the waiting arrivals in queue i) and global mean waiting time for a waiting customer in the whole system:

$$t_{w\,i} = w_{l\,i} / W_i \qquad (8a)$$

$$t_w = w_l / W \qquad (8b)$$

- Probability, that queue i is found at threshold $x_i = z$ and probability, that any queue is found at threshold $x_i = z$ :

$$p_{th\,i}[z] = \sum_{\underline{X}|x_i=z} p(\underline{X}) \qquad (9a)$$

$$p_{th}[z] = \sum_{i=1}^{g} q_i \cdot p_{th\,i}[z] \qquad (9b)$$

### 2.4 Implementation Aspects

Regarding the state dependent strategies and assuming a centralized dispatcher, a certain amount of system state information has to go back from the system to the dispatcher, dependent on the strategy. Fig. 4 shows a model, which describes this fact. Messages arriving with rate $\lambda$, have to be assigned to one of the queues, that means, they have to pass a dispatching service phase with service time $t_d$. Informations of the system state or its changes or blocking informations are sent back from the system or, respectively, from the ports of the queues through the communication system, e.g. a bus, to the dispatcher with rate $\lambda_u$, as far as they are needed for the dispatching strategy. These informations are assumed to pass an update service phase with service time $t_u$ in the dispatcher. So, the ne-

cessary mean utilization of the dispatcher for a given strategy can be estimated by $t_u \cdot \lambda_u + t_d \cdot \lambda$, and the additional overhead load for the communication system is determined by $\lambda_u \cdot h$, where h is the transfer time for a message of the communication system. As $\lambda_u$ is a function of the used strategy and its implementation, in the following the minimal necessary rate $\lambda_u$ is used as a comparison measure for the control overhead of a dispatching strategy. $\lambda_u$ is assumed to include informations on blocking, because losses usually are not allowed in switching controls.

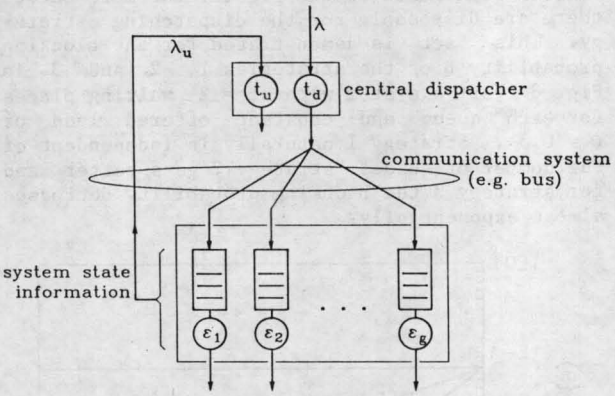


Fig. 4 Model for state dependent dispatching control

In the following some implementation methods and the appropriate rates $\lambda_u$ are given:

Strategy 1 and 2: Normally no system state information is necessary, but to avoid losses, for each blocked message one back message contributes to $\lambda_u$. Hence, it follows:

$$\lambda_u = \lambda \cdot B \qquad (10)$$

Strategy 3: As the arriving messages directly reach the dispatcher, only one information is necessary for each termination of a service phase in the system, to allow the dispatcher to reconstruct the system state. Therefore, the minimal necessary rate $\lambda_u$ is:

$$\lambda_u = Y \cdot \sum_{i=1}^{g} \varepsilon_i = \lambda \cdot (1 - B) \qquad (11)$$

Strategy 4: For each arrival, one information is necessary, whether the threshold th is reached, and for each termination of a service phase in the system, one information is necessary, whether the queue length falls below the threshold th. Hence, it follows:

$$\lambda_u = \lambda \cdot p_{th}[th-1] + \lambda \cdot (1 - B) \cdot p_{th}[th] \qquad (12)$$

Strategy 5: Each arrival at a full queue causes at the first attempt one blocking message. A queue is found full with the probability $p_{full} = p_{th}[z=n_i]$. For the second attempt, the strategies 5a, b and c must be distinguished:

a) If the second attempt is blocked, one more blocking message is necessary. Hence, it follows:

$$\lambda_u = \lambda \cdot (p_{full} + B) \qquad (13a)$$

b) After the first attempt, from each queue, which is not full, one information is necessary. Therefore, we have:

$$\lambda_u = \lambda \cdot p_{full} \cdot (1 + g \cdot (1 - p_{full})) \qquad (13b)$$

But the impementation method of strategy 4 can also be applied here with $th = n_i$.

c) After the first attempt, from each other queue one information is necessary, that means g messages all over. Thus, it follows:

$$\lambda_u = \lambda \cdot g \cdot p_{full} \qquad (13c)$$

## 2.5 Results

All considered strategies have been investigated in numerous parameter studies. Some typical examples of the obtained results are presented in the following.

First, the differences between the strategies should be illustrated by the global blocking probability B, the global mean waiting time $t_w$, and the waiting probability W versus the offered load, defined by $\rho = \lambda / \Sigma \varepsilon_i$. In the second example, the influence of the number of queues on the blocking probability B will be shown. Finally, the control overhead rate $\lambda_u$, as defined in section 2.4., is demonstrated for different strategies.

Fig. 5 shows the blocking probability B for a system with g = 4 queues, each with $s_i = 6$ waiting places. The strategies 1,2,3, 5a and 5b are considered. The dynamic strategy 3 is the best, while the random strategy 1 is the worst case. Thereby, strategy 3a and 3b have exactly the same results for the global characteristic values. In all investigated parameter studies, the blocking probability of the cyclic strategy 2 lies approximately in the middle between strategies 1 and 3 in logarithmic scale. Furthermore, two examples for strategy 5, namely 5a and 5b, are also shown in the diagram. The curves for different thresholds of the queue length of strategy 4, which are not included in the diagram, are located in the range between strategies 1 and 3, where every blocking level can be achieved through a proper choice of the thresholds.

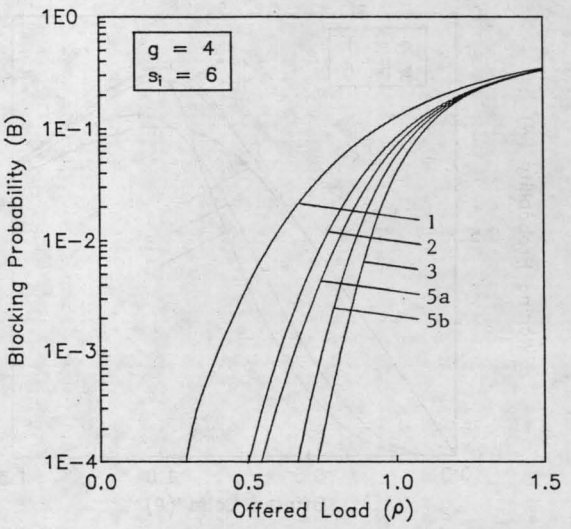


Fig. 5 Blocking probability versus offered load for dispatching strategies

Regarding the mean waiting time $t_w$ of the waiting customers for the same system, Fig. 6, the areas of normal load and overload must be separated. In the normal load range, strategy 3 is already the best, strategy 1 is the worst case, and strategy 2 lies between them. In the overload situation, the waiting time for strategy 3 is longer, because the throughput is higher in this case. Referring to the waiting time, the strategies 4 and 5 do not work very well, because intelligent dispatching is only active in the high load range. The examples of strategies 5a and 5b demonstrate this fact.

To get an idea of the mean waiting time of all arriving customers, the waiting probability W is shown in Fig. 7 for the same system. Due to the higher blocking probability in the overload case, the waiting probability decreases in this range.

The effect, that the cyclic strategy 2 is quite better than the random strategy 1 was already shown in other context in [7].

Summarizing all obtained results, the following statements can be given. The tendencies are approximately independent of the number of waiting places, that means, that the differences between the strategies remain relatively constant with increasing number of waiting places, although, of course, the blocking probability decreases and the waiting time increases. On the other hand, the difference between the strategies increases with the number of queues, but constant offered load per queue. This effect is the greater the more intelligent the strategies are, since the probability that an arriving customer finds a free place is the greater the more queues there are disposable for the dispatching strategy. This fact is demonstrated for the blocking probability B of the strategies 1, 2 and 3 in Fig. 8 for a system with $s_i = 2$ waiting places for each queue and constant offered load of $\rho = 0.5$ . Strategy 1 naturally is independent of the number of queues, strategy 2 gets better and for strategy 3 the blocking probability decreases almost exponentially.

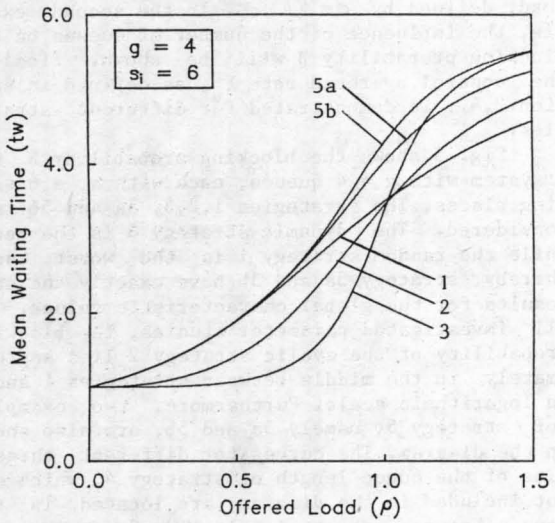


Fig. 6 Waiting time versus offered load for dispatching strategies

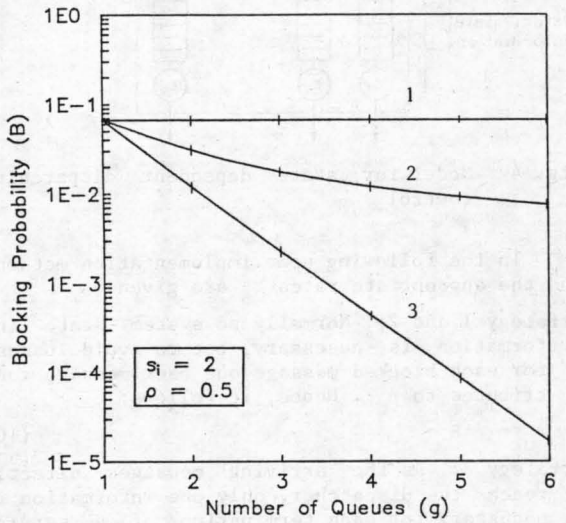


Fig. 8 Blocking probability versus number of queues for dispatching strategies

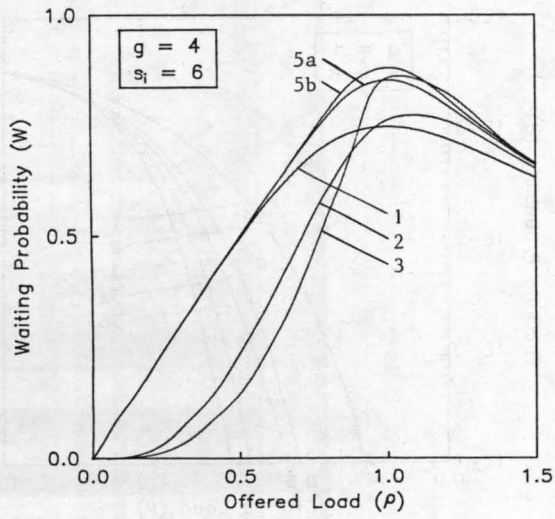


Fig. 7 Waiting probability versus offered load for dispatching strategies

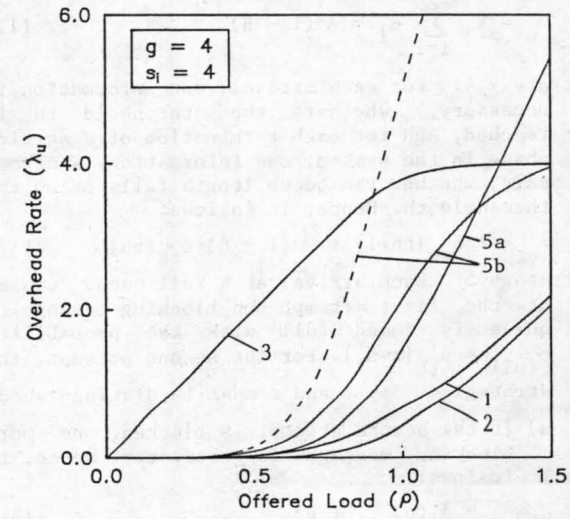


Fig. 9 Overhead rate versus offered load for dispatching strategies

Finally, the control overhead rate $\lambda_u$ will be discussed, which is shown in Fig. 9. In the normal load range, strategy 3 is the worst case, but $\lambda_u$ remains saturated at a relatively small level in the overload case, whereas the other strategies increase rapidly in the overload range. The non-intelligent strategies 1 and 2 naturally are the best strategies under the overhead aspect. For strategy 5b two curves are shown. The dashed curve stands for the application according to equation (13b), while the bold curve stands for the application according to equation (12), which also can profitably be implemented here.

## 3. PERFORMANCE OF POLLING STRATEGIES

### 3.1 Polling Strategies

As in chapter 2 for dispatching strategies, some corresponding polling mechanisms can be considered. Here, also many subcases could be regarded, but this is ommitted in the following list of examples, since the results do not differ significantly:

Strategy 1: Random polling, according to scheduling probabilities (e.g. equally distributed).

Strategy 2: Ordinary cyclic polling.

Strategy 3: Fully dynamic (state dependent) polling, that means, that the longest queue is the next queue being served.

Strategy 4: Partially dynamic (state dependent) polling, that means according to, e.g., strategy 1, if the queue lengths are below a given threshold. But if a queue exceeds this threshold, the appropriate queue is the next queue being served.

Strategy 5: Like strategy 4, but with an hysteresis range of the queue length instead of one fixed threshold.

### 3.2 Basic Queueing Model

Fig. 10 shows the generic queueing model for the investigations of polling strategies. Thereby, g queues are polled by the polling server with the service rate $\varepsilon$ corresponding to the scheduling strategy. A queue i has $s_i$ waiting places and an arrival rate $\lambda_i$. The arrival and service processes are supposed to be of Markovian type, because of the same reasons as in the model for dispatching mechanisms.

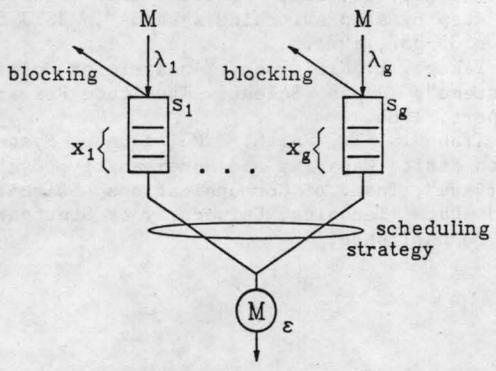


Fig. 10 Basic model for polling strategies

### 3.3 Solution Method

As the solution is very similar to that for dispatching strategies, a detailed description can be ommitted here.

The state variables are the numbers $x_i$ of occupied places in each queue. One additional state for the case of a free server must be used. For strategy 2 one more state variable is necessary for the cycle number. For strategy 5 one more state variable is necessary for each queue to indicate between the hysteresis thresholds, whether at last the upper limit has been exceeded or the lower limit has been crossed.

### 3.4 Results

Although polling mechanisms have also been investigated in numerous parameter studies, only one example will be shown here, because the differences between the strategies are very small.

Fig. 11 shows the blocking probability B and Fig. 12 the mean waiting time $t_w$ of the waiting customers for a symmetrical system with g = 4

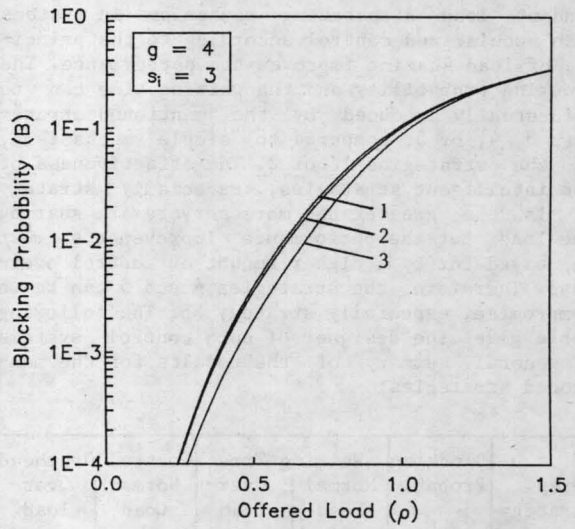


Fig. 11 Blocking probability versus offered load for polling strategies

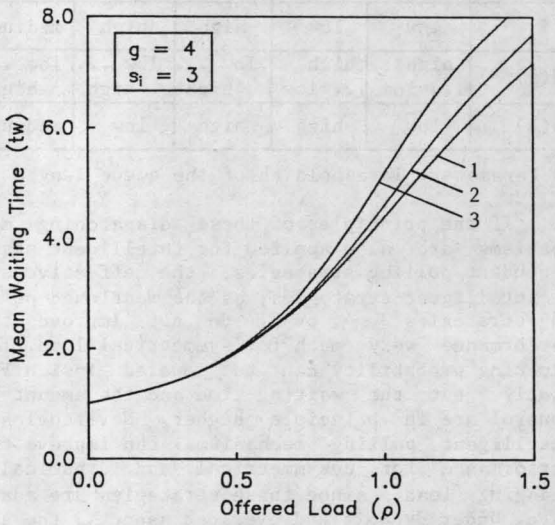


Fig. 12 Waiting time versus offered load for polling strategies

queues, each with $s_i = 3$ waiting places, versus the offered load $\rho \triangleq \Sigma\lambda_i / \varepsilon$. The curves show that the cyclic strategy 2 has almost the same blocking probability as the random strategy 1, while the dynamic strategy 3 is insignificantly better. The waiting time is almost the same for all strategies in the normal load range, but in the overload range the intelligent strategies are worse.

All cases for different thresholds of strategies 4 and 5 lie between strategies 1 and 3, and a comparison between strategy 4 with a threshold th and strategy 5 with an hysteresis range around th renders almost the same results.

So, it seems to be not very usefull to implement state dependent polling strategies, but the advantage must be seen in the behaviour for unsymmetrical load and momentary overload, since a state dependent strategy automatically includes dynamic priorities and therefore adapts itself to the queue-individual loads. Investigations on these effects are in work at present.

## 4. CONCLUSIONS

The implementation of intelligent state dependent load dispatching mechanisms in systems with modularized control according to the principle of load sharing improves the performance. The blocking probability and the waiting time can be evidentually reduced by the mentioned strategies 3, 4, or 5, compared to simple mechanisms, as the strategies 1 or 2. The effectiveness of the intelligent strategies, especially strategy 3, is the greater the more servers are sharing the load, but the performance improvements must be payed for by a higher amount of control overhead. Therefore, the strategies 4 and 5 can be a compromise, especially strategy 5b. The following table gives the designer of such control systems a general summary of the results for the mentioned strategies:

| Disp. Strategy | Blocking Probab. | Waiting Time | | Control Overhead | |
|---|---|---|---|---|---|
| | | Normal Load | Overload | Normal Load | Overload |
| 1 | high | high | low | low | low |
| 2 | medium | medium | medium | low | low |
| 3 | low | low | high | high | medium |
| 4(a) *) | high ... low | high ... low | low ... high | low ... high | low ... high |
| 5(b) | low | high | high | low | medium |

*) Parameter: Threshold th of the queue length

If the principles of these dispatching mechanisms are also applied for intelligent state dependent polling strategies, the effectiveness of intelligent strategies, as the mentioned polling strategies 3, 4, or 5, do not improve the performance very much for symmetrical load. The blocking probability can be lowered insignificantly, but the waiting time and the amount of control are in principle higher. Nevertheless, intelligent polling mechanisms can improve the performance for unsymmetrical and dynamically changing load, since these strategies are adaptive. Under dynamic and overload aspects, the application can although be profitable, but these effects are not investigated in this paper.

For the realization of intelligent scheduling strategies, a centralized control unit, e.g., a bus control unit, is suitable. The overhead of the control information which has to be transferred from the single modules to this scheduling control unit, has been estimated in this paper for the various dispatching mechanisms. But a decentralization of the scheduling control is also imaginable, e.g., being located in a microprocessor control of the communication system access ports of each unit. In this case, the scheduling control information messages with a fixed destination must be replaced by broadcast messages.

Furthermore, analytic performance investigations of the whole system can be facilitated by the implementation of the best scheduling strategies, because then certain approximations can be applied. So, e.g., the blocking probability for the dispatching strategy 3 can be approximated quite well by the blocking probability of a single queue M/M/g-s delay-loss system with $s = \Sigma s_i$ waiting places.

## ACKNOWLEDGEMENT

The author would like to express his gratitude to Prof. P. Kuehn, Head of the Institute of Communications Switching and Data Technics, University of Stuttgart, for supporting this work.

## REFERENCES

[1] W. Denzel, W. Weiss, "Modelling and Performance of a New Integrated Switching System for Voice and Data with Distributed Control", Proc. 10th Int. Teletraffic Congress, Montreal, paper 1.1-3, 1983.

[2] P. Kuehn, "Tables on Delay Systems", Inst. of Communications Switching and Data Technics, University of Stuttgart, 1976.

[3] L. Kleinrock, "Queueing Systems", Vol. I/II, Wiley + Sons, New York, 1975/76.

[4] R. Schassberger, "Ein Wartesystem mit zwei parallelen Warteschlangen", Computing 3, pp. 110-124, 1968.

[5] U. Herzog, P. Kuehn, "Comparison of some Multiqueue Models with Overflow and Load-Sharing Strategies for Data Transmission and Computer Systems", Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, pp. 449-472, 1972.

[6] G. Császár, R. Konkoly, T. Szádeczky-Kardoss, "Optimal Allocation of Calls to Private Automatic Branch Exchange Operators", Budavox Telecommunication Review, Budapest, Vol. 4, pp. 1-6, 1982.

[7] M. Buchner, S. Neal, "Inherent load balancing in step by step switching systems", BSTJ 50, pp. 135-357, 1971.

[8] H. Takagi, L. Kleinrock, "Analysis of Polling Systems", Japan Science Institute Research Report, 1985.

[9] P. Tran-Gia, T. Raith, "Multiqueue Systems with Finite Capacity and Nonexhaustive Cyclic Servive", Inst. of Communaications Switching and Data Technics, University of Stuttgart, to be published.