Performance Analysis of a Multiprocessor System Model with Process Communication

I. F. AKYILDIZ
College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332, USA

This work analyses a multiprocessor system in which processes communicate with each other via buffers using SEND and WAIT operations. Since the buffers are of finite capacity, a process which cannot execute a synchronisation operation successfully must leave the processor and go into the blocking state. The analysis of the multiprocessor system is executed hierarchically using two models – (Global Model and Process Communication Model). The interprocess communication is analysed by using the process communication model, a closed queueing network model with finite station capacities. An analytical method is developed for the solution of the process communication model. The method provides exact results for two-station cases and accurate approximate results for multiple station cases. The blocking probabilities and the blocking times of processes are computed from the process communication model and are used as input parameters for the global model which can then be solved by appropriate existing product form network methods. The performance measures such as utilisation, throughput, mean response time, mean queue length, and in particular, the blocking overhead due to the process communication, are obtained. The analytical results are validated by simulation of the entire system.

Received March 1989, revised September 1991

1. INTRODUCTION

There is an increasing scientific and commercial interest in the investigation of multiprocessor hardware and software. Multiprocessor systems are a special class of distributed computing systems that appear to represent the most promising way of obtaining the high performance computers needed in many application fields, such as expert systems, artificial intelligence and large scale system simulation. Characteristics such as fault tolerance, flexibility, functional upgrading and cost effectiveness are other motivations that have spurred the realisation of multiprocessor systems. To pursue these goals, a variety of multiprocessor architectures with different design alternatives have been proposed, implemented and made commercially available but their relative merits are not yet fully understood. It is thus very important to develop methodologies and tools for the prediction of the performance of multiprocessor architectures, so that system designers can verify how well different alternatives suit certain given performance specifications.

A model of a multiprocessor usually consists of two parts: the description of the architecture and the definition of the workload under which the performance predictions should be obtained. The key elements in the model development are the choice of the level of abstraction used to describe the system, the selection of the system features to be included in the model, the assignment of numerical values to the model parameters and the definition of appropriate performance indices. The choice of the system features to be included in the model is a very important step in the modelling process since it must ensure the adequacy of the description without introducing unneeded complexity. A class of models that is widely used for performance prediction and performance analysis is based on the theory of queueing networks. Several researchers applied successfully queueing network models to analyse multiprocessor systems.[11] To overcome the computational complexity of the exact queueing model for the performance analysis of large-scale multiprocessor systems, many approximate methods have been introduced for synchronous[6, 8, 13] and asynchronous[9, 10, 11] systems.

Marsan and Gerla[9] use queueing network models to analyse the performance of asynchronous systems. Since the number of states for their model increases rapidly with the system size, they reduced the size of the Markov chains by an approximate lumping technique on the assumption that the lumped process still satisfies the Markov property. They introduced four different approximate models and compared these models with the exact one for various multiprocessor configurations. For some paradigms, they found that the approximation error is at most about 10%. Marsan *et al.*[10] also applied generalised stochastic Petri Nets (GSPN) to the performance evaluation of multiprocessor systems. They show that GSPN's are equivalent to continuous-time stochastic processes.

In multiprocessor systems, coordination problems between the processes can occur. It is clear that the system overhead due to the interprocessor communication can be significant and must be taken into account. In this work we are modelling the synchronisation of processes in multiprocessor systems. Results show, as intuitively expected, the effects of the synchronisation may induce a much more significant performance reduction when the communication load is high.

The paper is organised as follows: in Section 2 we describe the characteristics of the multiprocessor system to investigate. In Section 3 two queueing network models (global and process communication models) are developed in Section 4 algorithms are presented for the analysis of the process communication model which is a closed queueing network model with finite station capacities. Section 5 contains three paradigms which demonstrate, as intuitively expected, the effect of process communication on the performance of multiprocessor systems. Finally Section 6 concludes the paper and suggests some avenues for further research.

2. DESCRIPTION OF THE MULTIPROCESSOR SYSTEM (SYSTEM MODEL)

The multiprocessor system to investigate has the following structure:

(i) The system consists of m identical processors.
(ii) The number of processes circulating in the system is fixed at K. All processes belong to the same class.
(iii) The scheduling discipline for the process servicing is PS (processor sharing).
(iv) There is a global memory which is sufficiently large so that no queues are necessary for entering.
(v) The processes in the system can be in one of the three states, i.e. ready, active, blocked.

So we obtain the following classical state diagram for processes:

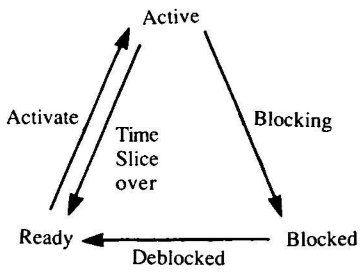

Figure 1. State diagram of the process.

In the following, we explain the process states and state transitions in detail. In the system, we distinguish between two kinds of queues.

(a) *Ready Queue*

This queue manages the physical allocation of processors. Processes are waiting in this queue in order to be serviced by the processors. The scheduling discipline is PS (processor sharing). An idle processor always serves the first of processes ordered in the queue. In Fig. 1, the Ready Queue represents the Ready State.

(b) *Communication Queues*

In multiprocessor systems, it is common that the processes communicate with each other in order to reach a synchronisation or to exchange messages with each other. In this system, there are so-called communication queues (CQ) (buffers) for this purpose. Each process has its own finite buffer, i.e. communication queue. In Fig. 1, the communication queues represent the blocked state. Each process can send a message to any other process or receive a message from any other process. Note that we assume that the system provides the deadlock free mechanisms.

The sending of a message can be executed by using the synchronization operation:

SEND (Message, Destination)

A process can receive a message from its own buffer by using the operation:

WAIT (Message)

The sending or receiving of a message by a process is possible if the corresponding process is active, i.e. if it is running on one of the processors. The messages are processed by FCFS discipline. After these explanations, Fig. 1 can be extended to the following form:

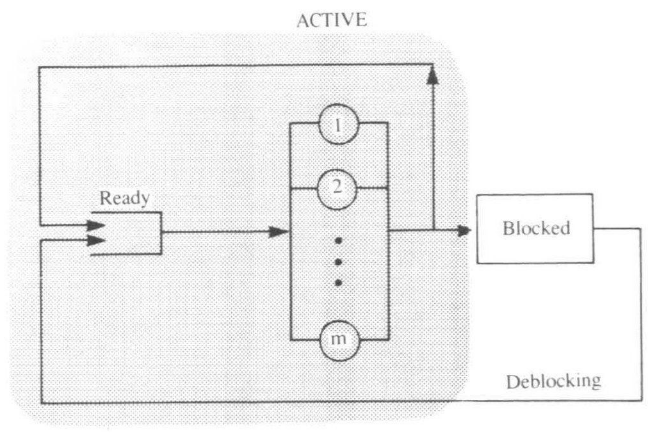

Figure 2. Detailed system model.

As it can be seen in Fig. 2, a process leaves the processor when

(i) the time slice is over. In this case it goes back to the ready-state.
(ii) a synchronisation action, i.e. SEND or WAIT operation, is not executable. In this case it proceeds to a blocked state. After some residence time in the blocked state, the process deblocks and returns to the ready state.

The state transitions are explained in detail by using the following formal definition.

Definition 1

A process $k(p\_k)$ (for $k = 1, \ldots, K$) can change its state due to the following conditions:

$$Active \rightarrow Blocked = \begin{cases} SEND\,(message, p\_n) & \text{if} \quad CQ\_n \text{ is full} \\ WAIT\,(message) & \text{if} \quad CQ\_k \text{ is empty} \end{cases}$$

$$Blocked \rightarrow Ready = \begin{cases} WAIT\,(message) & \text{if} \quad p\_n \text{ is full} \\ SEND\,(message, p\_n) & \text{if} \quad p\_k \text{ executes} \end{cases}$$

$$Active \rightarrow Ready \quad \text{if} \quad \text{the time slice is over.}$$

The informal interpretation is: a process goes from active to blocked state, if it sends a message to any other process whose buffer is already full. In this case, the process will be deblocked (i.e. from blocked to ready state), if the corresponding process *p_n* executes a WAIT-operation and receives a message so that there will be a space available for the message of the blocked process *p_k*. On the other hand, a process *p_k* can be blocked if it executes a WAIT-operation, i.e. it wants to receive a message from its own buffer which is empty. The process *p_k* will then be deblocked, if any other process sends a message to it.

In this work, we analyse this system and obtain performance measures: utilisation of the processors, system throughput, mean response time, mean number of processes in the system, mean queue lengths, in particular, the blocking times and blocking probabilities of the processes, in other words, the effect of process communication (the overhead) on the performance of the system. The analytical results will be validated by simulation of the system.

3. GLOBAL QUEUEING NETWORK MODEL

The following global queueing network model is developed for the described system:

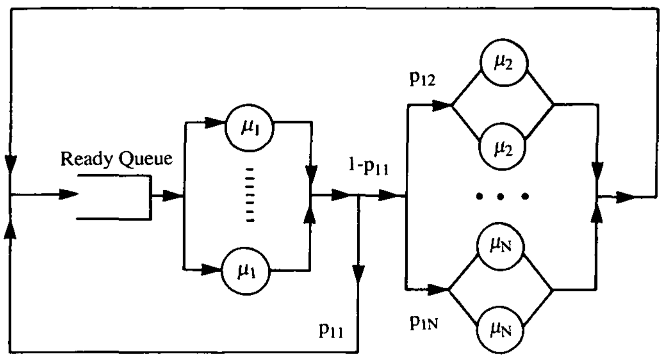

Figure 3. The global queueing network model.

The global queueing network model has the following characteristics: There are fixed K circulating processes and N stations where the first station represents the identical processors and the global memory. The stations 2 to N represent the blocked state. The stations 2 to N are of Type 3[4] (delay stations; no queue in front of the servers). So we obtain as many delay stations as there are processes in the system, i.e.

$$(N-1) := K$$

(*Total Number of Delay Stations*):
= (*Total Number of Processes*)

The service time of a process in the ith station is exponentially distributed with mean value $1/\mu_i$ for $i = 1, \ldots, N$. Note that the mean service time $1/\mu_i$ (for $i = 2, \ldots, N$) of the delay stations is equal to the blocking times of each process because of communication, i.e.

$$\frac{1}{\mu_i} := \bar{t}_{B_i} \quad \text{for} \quad i = 2, \ldots, N \tag{1}$$

where

$\frac{1}{\mu_i}$ is the mean service time of the delay stations which is unknown yet

$\bar{t}_{B_i}$ is the mean blocking time of the process i (for $i = 2, \ldots, N$)

As it can be seen in Fig. 3 the processes run through the stations in a relatively simple network. We must, of course, consider the effect of process communication in this model. One influence is already considered by setting the blocking times equal to the mean service times of the delay stations. An additional influence is considered in the transition probabilities of processes. As mentioned before, two ways are possible, if a process leaves one of the processors: either it goes back to the ready queue (this is the transition probability p_{11}), or it proceeds with p_{1j} (for $j = 2, \ldots, N$), to one of the delay stations because of non-executable synchronisation action.

The transition probability p_{11} that a process leaves the processor and returns back to the ready queue because the time slice is over, is given by:

$$p_{11} = 1 - \sum_{i=2}^{N} \frac{P_{B_i}}{N-1} \tag{2}$$

where P_{B_i} is the probability that the ith process is blocked.

The transition probabilities from the processors to the delay stations, p_{1j} (for $j = 2, \ldots, N$), are computed from the ratio of the blocking probability P_{B_i} of each station and the number of processes $(N-1)$ in the system.

$$P_{1j} = \sum_{i=2}^{N} \frac{P_{B_i}}{N-1} \frac{P_{B_i}}{\sum_{i=2}^{N} P_{B_i}} = \frac{P_{B_i}}{N-1} \tag{3}$$

If a process is deblocked, it goes back to the processor queue with the transition probability $p_{j1} = 1$ for $j = 2, \ldots, N$. Since the transitions between the delay stations are not possible, the remaining transition probabilities p_{ij} for all $i = 2, \ldots, N$ and $j = 2, \ldots, N$,are 0.

The major problem in the global queueing network model is the computation of the transition probabilities (which correspond to the blocking probabilities) and mean service times of the delay stations (which correspond to the blocking times) somehow, so that they can be used as input parameters.

4. PROCESS COMMUNICATION MODEL

In order to compute the mean blocking times $\bar{t}_{B_i}$ and the blocking probabilities P_{B_i} of the processes we have developed the process communication model and made the following assumptions: Processes are abstract servers and there are K processes in the model. Each process i has its own finite buffer M_i. A process with a finite buffer represents here a station. The messages are the jobs (customers) which are serviced by the processes after FCFS discipline. The number of messages is fixed at C.

$$\sum_{i=1}^{K} c_i = C$$

where c_i is the number of messages in buffer and in service of the ith station.

The mean service time of the station corresponds to the time proportion between the WAIT and SEND operation of each process from the system description. This service time is exponentially distributed with mean service rate μ_i for $i = 1, 2, \ldots, K$. We also assume that the transition probabilities of messages between the stations is given by p_{ij}. This model is shown in Fig. 4.

One of the most important problems to realise regarding blocking queueing networks is that finite station capacities and blocking can introduce the problem of system deadlock. Deadlock may occur if a message which has finished its service at station i's server wants to join station j whose capacity is full. That message is blocked in station i. Another message which has finished its service at jth station now wants to proceed to the ith

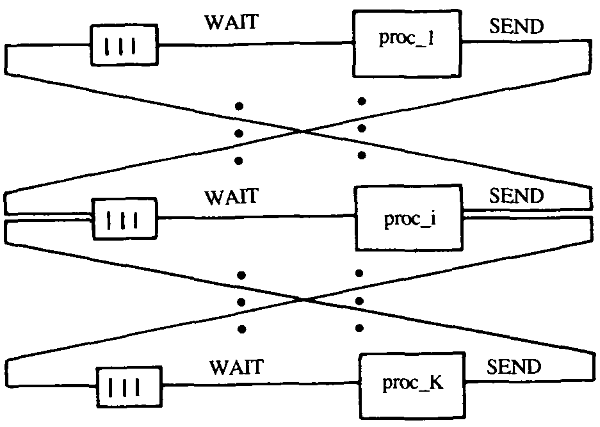

Figure 4. Process communication model.

station whose capacity is also full. It blocks station j. Both messages are waiting for each other. As a result a deadlock situation arises. The following assumption states that a closed queueing network containing finite station capacities deadlock-free if and only if for each cycle CYC in the network the following condition holds:[7]

$$C = \sum_{j \in CYC} M_j$$

Simply stated, the total number of messages in the network must be smaller than the sum of station capacities in each cycle.

The system state **c** is defined

$$\mathbf{c} = (c_1, c_2, \ldots, c_K)$$

where c_i for $i = 1, 2, \ldots, K$, is the number of messages in the ith station.

A state **c** is *feasible*, if

$$\sum_{i=1}^{K} c_i = C \wedge (c_i < M_i \wedge c_i > 0)$$

for $i = 1, 2, \ldots, K$.

In other words, a state is feasible, if the sum of the number of messages in each station is equal to the total number of messages in the network and the number of messages does not exceed the capacity of the station ($c_i < M_i$) so that the SEND operation can be executed, or there is at least one message in the station ($c_i > 0$) so that the WAIT operation can be executed.

In the system description model, we have assumed that the processes can execute a synchronisation action if they are active, i.e. if they possess the processors. In the developed process communication model all processes can execute a synchronisation action as if they were active. Since with PS (processor sharing) scheduling discipline, all processes can be active simultaneously, independent of the number of processors, the concept is correct that the blocking probabilities and blocking times can be obtained from the process communication model and can be used as input parameters for the global model. The process communication model is a closed queueing network model with finite station capacities where blockings may occur. Since blocking causes interdependencies between stations, such networks cannot be analysed by existing product form algorithms.[5] In recent years there has been an increased interest in the analysis of queueing networks with blocking. This is probably due to the realisation that these queueing networks are useful in modelling computer systems, communication networks, and flexible manufacturing systems. Recently a special issue[3] appeared in a journal which give the state-of-the-art in this research area.

We have two options to take: either we simulate the process communication model and get the blocking probabilities and blocking times and use them in the global model which could be then solved analytically (a kind of hybrid simulation study), or try to find a new method for the analysis of queueing networks with blocking so that the blocking probabilities and mean blocking times can be obtained analytically (a pure analytical investigation). We select the latter.

The following is a brief description of our two algorithms. The first algorithm[1] provides exact results for two-station queueing networks with blocking. The second algorithm[2] provides accurate approximate results for performance measures in multiple station queueing networks with blocking.

4.1 Exact Solution for Two Station Networks with Blocking

The solution is based on the following idea which is described in detail in Ref. 1. The state space of a closed queueing network model with blocking can be reduced by considering the finite station capacities. The immediate neighbours of the feasible states are the blocking states of the network. An equivalent queueing network without limitation of station capacities can be found with appropriate total number of messages which provides exactly the same state space structure as the blocking queueing network. Markov processes describing the evolution of both networks over time have the same structure. Since the equivalent closed queueing network without blocking is a product form network, exact results can easily be computed for queue length distributions of the blocking network. From these observations we can derive exact formulas for performance measures in two-station queueing networks with blocking.

The *normalisation constant* is a $(C+1)$ dimensional vector with

$$\mathbf{G} = \begin{bmatrix} G(0) \\ G(1) \\ \cdots \\ \cdots \\ G(C) \end{bmatrix}$$

with $G(0) = 1$ and is computed from:

$$\mathbf{G} = \mathbf{Y}_1 * \mathbf{Y}_2 \tag{4}$$

where $*$ is the convolution operation and Y_i (for $i = 1, 2$) is a $(C+1)$-dimensional vector with

$$Y_i = \begin{bmatrix} y_i(0) \\ y_i(1) \\ \cdots \\ \cdots \\ y_i(C) \end{bmatrix}$$

where the components $y_i(.)$ are defined as follows:

$$y_i(c) = \begin{cases} 1 & \text{if} \quad c = 0 \\ \dfrac{y_i(c-1)}{\mu_i} & \text{if} \quad c = 1, 2, \ldots, M_i + 1 \\ 0 & \text{if} \quad c = M_i + 2, \ldots, C \end{cases}$$

The *total throughput* of the blocking network is computed from:[1]

$$\lambda_B(C) = \frac{G(C-1)}{G(C)} \tag{5}$$

The *throughput* of each station is computed by

$$\lambda_i(C) = e_i \lambda_B(C) \tag{6}$$

where e_i is the mean number of visits of messages to station i and is computed by

$$e_i = \sum_{j=1}^{N} e_j p_{ji} \tag{7}$$

Note that $e_i = 1$ for two-station cyclic networks.

The *mean number of messages* in the ith station is computed by

$$\bar{c}_i(C) = \frac{1}{G(C)} [D_i + E_i + F_i] \tag{8}$$

where

$$D_i = \sum_{n=C-M_j}^{M_i} n y_i(n) y_j(C-n)$$

$$E_i = M_i y_i(M_i + n) y_j(C - M_i - l)$$

$$F_i = (C - M_j) y_i(C - M_j - n) y_j(M_j + n)$$

for $i, j = 1, 2$ and $i \neq j$. The informal interpretation of this formula is that D_i includes the feasible states where E_i and F_i include the blocking states.

The *mean queue length* of the ith station is computed from:

$$\bar{Q}_i(C) = \frac{1}{G(C)} T_i \tag{9}$$

where

$$T_i = \sum_{n=C-M_j}^{M_i} \alpha(n-1) y_i(n) y_j(C-n)$$

$$+ M_i - 1 y_i(M_i + 1) y_j(C - M_i - 1)$$

$$+ \alpha(C - M_j - 1) y_i(C - M_j - 1) y_j(M_j + 1)$$

for $i, j = 1, 2$ and $i \neq j$, with the auxiliary function

$$\alpha(n) = \begin{cases} 0 & \text{if} \quad n \leqslant 0 \\ n & \text{if} \quad n > 0 \end{cases}$$

The *mean response time of messages* (time spent by a message in queue, in service and in blocking phase) and the *mean waiting time of a message* in the ith station are computed from Little's Law:

$$\bar{t}_i(C) = \frac{\bar{c}_i(C)}{\lambda_i(C)}; \; \bar{w}_i(C) = \frac{\bar{Q}_i(C)}{\lambda_i(C)} \tag{10}$$

for $i = 1, 2$.

The most important formulas for our study are the *blocking probabilities* and the *mean blocking times* for each process so that we can use them as input parameters for the global model.

The *blocking probability*, i.e. the probability that the ith station is in a blocked state is computed from

$$P_{B_i}(C) = \frac{1}{G(C)} y_i(C - M_j - 1) y_j(M_j + 1) \tag{11}$$

The *mean blocking time* of the ith station is determined from

$$\bar{t}_{B_i}(C) = \bar{t}_i(C) - \bar{w}_i(C) - 1/\mu_i \tag{12}$$

where $\bar{t}_i$ and $\bar{w}_i$ are computed from equation (10).

4.2 Mean Value Analysis for Blocking Queueing Networks

Mean value analysis[12] (in short form MVA) for infinite capacity queueing networks enjoys a wide applicability in performance evaluation of computer systems, communication networks and flexible manufacturing systems since the last decade. It is a very easy and fast algorithm for product form queueing network models. The stepwise behaviour of the MVA permits to determine the blocking events in networks with finite station capacities. Two basic characteristics of blocking network models must be considered in the algorithm: A station whose successor station capacity is full is blocked and a station whose capacity is full cannot accept any job. The fact that a job cannot join another station with a full capacity has the effect of increasing the mean response time of the source station. The job blocks the source station until a place is available in the destination station. This place will be available after a job has finished service at the full station. Accordingly, the mean response time of the jobs in the blocked station j increases by the mean remaining service time (mean residual time) BZ_i of the destination station i:[2]

$$\bar{t}_j(k) = \frac{1}{\mu_j} [1 + \bar{k}_j(k-1)] + BZ_i \tag{13}$$

In Ref. 2 we demonstrated that the mean blocking time is equal to the mean remaining service time (mean residual service time) for exponentially distributed service times:

$$BZ_i = \frac{1}{\mu_i} \tag{14}$$

for $i = 1, \ldots, N$.

If a source station j has many successor stations and one of them is full, the mean response time of the source station j increases by the mean remaining service time of the full station BZ_i multiplied by the transition probability by which the job would proceed to the full station i weighted by the ratio of the mean number of visits of the full station j, e_j, to the mean number of visits of the blocked station i, e_i:

$$\bar{t}_j(k) = \frac{1}{\mu_j} [1 + \bar{k}_j(k-1)] + BZ_i \left\{ \frac{p_{ij} e_j}{e_i} \right\} \tag{15}$$

for $i = 1, \ldots, N$.

The second general characteristic of blocking queueing networks is that a full station cannot accept a new job. In other words, that job does not join the full station. As a result, the mean response time of the full station i is computed by the mean service time of jobs which are already in the station i:

$$\bar{t}_i(k) = \frac{1}{\mu_i}[\bar{k}_i(k-1)] \qquad (16)$$

We start the computation with classical mean value analysis[12] and compute the mean response time from (15) by setting the BZ_i to zero. The *throughput* is determined from:

$$\lambda(k) = \frac{k}{\sum_{i-1}^{N} e_i \bar{t}_i(k)} \qquad (17)$$

and the *mean number of jobs* in station i from:

$$\bar{k}_i(k) = \lambda(k) e_i \bar{t}_i(k) \qquad (18)$$

After each iteration we will check to guarantee that the mean number of jobs in each station is less than or equal to the capacity of that station (i.e. $\bar{k}_i < B_i$), for all $i = 1, \dots, N$. If the capacity of a station is exceeded in an iteration, we repeat the iteration with the new suggested formulas, (15) and (16). If an additional blocking event occurs in the same station, the mean response time of the destination station remains the same while the mean response time of the blocked station is again increased. This is repeated until the total number of jobs K is reached in the iteration.

The *blocking probability* of a station is computed from the proportion of the mean blocking time to the mean response time of jobs in a station:[2]

$$P_{B_i}(k) = \frac{BZ_i(k)}{\bar{t}_i(k)} \qquad (19)$$

for $i = 1, \dots, N$. The *blocking time* of the ith station is the final value of $BZ_i(K)$.

REMARK: *As we demonstrated in Ref. 2 this approach is very fast and provides fairly accurate results. Since it is not the main purpose of this work to treat in detail the methods for queueing networks with blocking, we refer to Ref. 2 for the evaluation of this method.*

As mentioned before we can compute the mean blocking times from equation (15) and the blocking probabilities from equation (19) respectively, for each process and use the results as input parameters for the global model which is then analysed by existing product form algorithms. In the following section we give some numerical examples.

5. NUMERICAL EXAMPLES

5.1 Example 1

The example system consists of $m_1 = 1, 4, 8, 16, 20$ processors. The service time of the processors is exponentially distributed with mean value $1/\mu_1 = 61$ sec. The scheduling discipline of the processors is PS (processor sharing). There are $K = 20$ processes circulating in the system. Each process communicates only with another process, i.e. we obtain 10 process pairs for the communication. The buffers for message change of the processes are finite and have the capacities $M_i = 10$ for $i = 1, 2, \dots, 20$. There are total $C = 11$ messages per process pair. It is assumed that the processes have the following mean values for communication times:

Table 1

Process pair	$1/\mu_i$	$1/\mu_j$
1	9.69	9.26
2	10.30	9.95
3	9.91	9.90
4	9.92	9.85
5	10.34	9.26
6	10.79	9.06
7	10.78	10.11
8	9.19	9.72
9	9.52	11.08
10	11.17	9.13

This given multiprocessor system is analysed hierarchically. The process communication model consists of 10 two-station closed queueing network models each representing a pair of processes communicating with each other. By using the exact solution suggested in Section 4.1 we compute the blocking probabilities, equation (11), and mean blocking times, equation (12), which are listed with simulation results in Table 2:

Table 2

	Blocking Probabilities			Mean Blocking Times		
Pair	An.	Sim.	Dev.	An.	Sim.	Dev.
1	0.000	0.000	0.000	0.000	0.000	0.000
2	0.491	0.489	0.000	0.431	0.428	0.000
3	0.005	0.000	0.000	0.000	0.000	0.000
4	0.472	0.485	2.700	0.348	0.354	1.700
5	0.032	0.000	0.000	0.000	0.000	0.000
6	0.041	0.036	14.000	0.000	0.011	0.000
7	0.009	0.000	0.000	0.000	0.000	0.000
8	0.357	0.430	17.000	0.068	0.074	7.400
9	0.014	0.000	0.000	0.000	0.000	0.000
10	0.476	0.495	3.800	1.082	1.078	0.000
11	0.013	0.000	0.000	0.000	0.000	0.000
12	0.375	0.497	24.500	1.729	1.734	0.300
13	0.009	0.000	0.000	0.000	0.000	0.000
14	0.432	0.492	12.200	0.665	0.677	1.800
15	0.424	0.491	13.600	0.527	0.527	0.000
16	0.006	0.000	0.000	0.000	0.000	0.000
17	0.466	0.496	6.000	1.551	1.558	0.000
18	0.046	0.000	0.000	0.000	0.000	0.000
19	0.024	0.000	0.000	0.000	0.000	0.000
20	0.554	0.497	11.500	2.040	2.043	0.000

The global model consists of $N = 21$ stations (the first station represents the processors and the remaining stations 2 to 21 the blocked states). The mean service times of the stations 2 through 21 are obtained from mean blocking times which computed from the process communication model. The transition probabilities are computed from equations (2) and (3). The analytical and simulation results for performance measures are depicted in Figs 5a, b, c, d.

5.2 Example 2

The system has $m = 1, 2, 4$ processors and $K = 4$

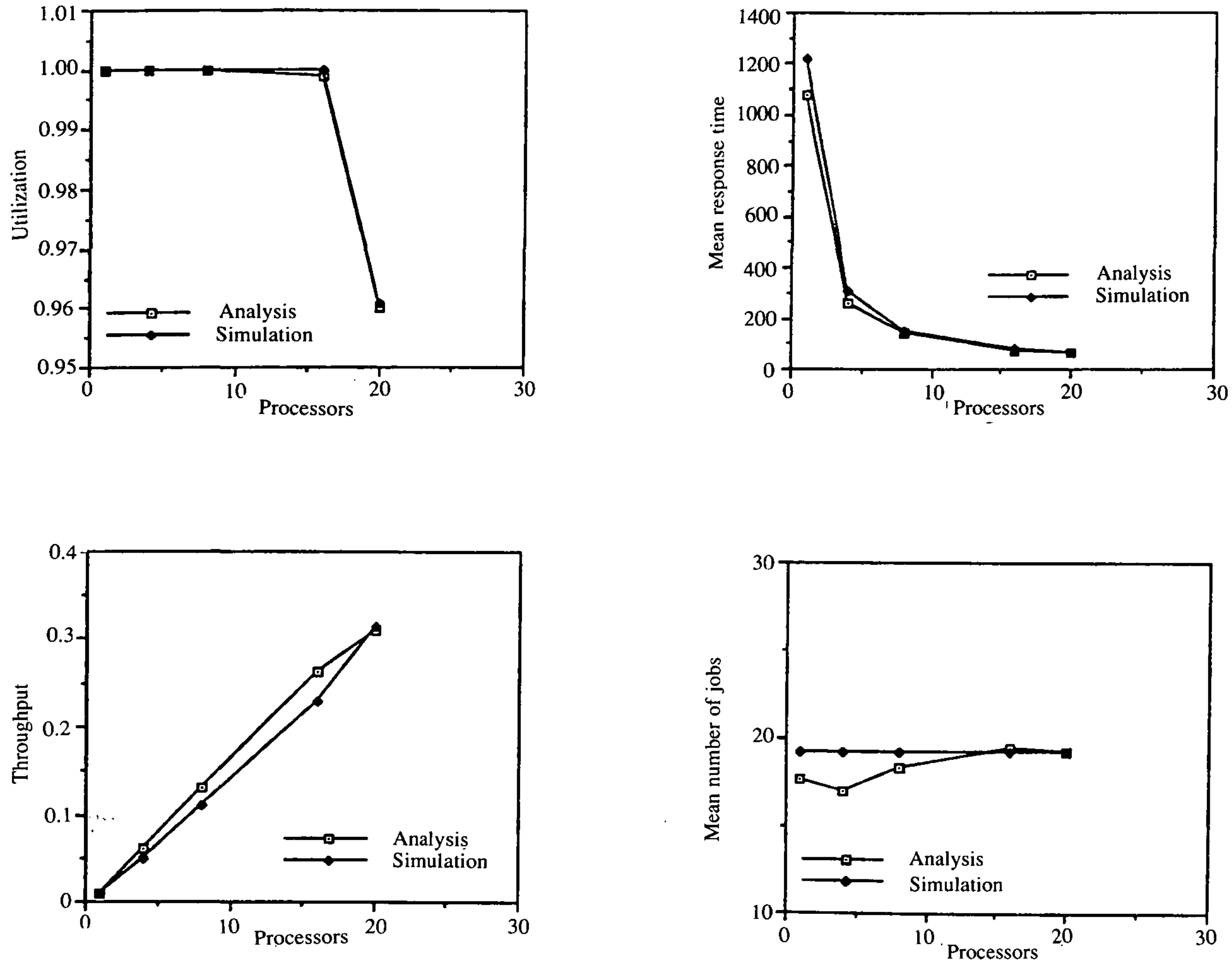

Figure 5. (a) Utilisation. (b) Mean response time. (c) Throughput. (d) Mean number of jobs.

processes. The processor service time is exponentially distributed with mean value $1/\mu_1 = 59$ sec. The processes communicate with each other with uniform probability. Each process has its own buffer. The buffers for the communication are finite of $M_i = 4$ (for $i = 1, 2, 3, 4$) capacities. The total number of messages which can be sent or received is $C = 14$. The mean message communication times are given in Table 3:

Table 3

Process no.	$1/\mu$
1	12.92
2	5.56
3	6.18
4	9.04

We develop a process communication model with 4 stations of finite capacities ($M_i = 4$) which is investigated by the method of Section 4.2. In Table 4 we see the analytical and simulation results.

The computed mean blocking times and blocking probabilities are used as input parameters for the global model which consists of 4 stations where the first station represents $m = 1, 2, 3, 4$ processors and the stations 2, 3, 4 model the blocked state. We obtain the following results which are plotted in Figs 6a, b, c, d.

Table 4

	Blocking Probabilities			Mean Blocking Times		
Process no.	An.	Sim.	Dev.	An.	Sim.	Dev.
1	0.001	0.000	0.000	0.000	0.000	0.000
2	0.589	0.623	5.731	3.852	4.021	4.400
3	0.342	0.332	2.922	7.456	7.295	2.152
4	0.361	0.372	3.043	9.647	9.906	2.682

5.3 Example 3

The multiprocessor system consists of $m = 1, 2, 4, 8, 10$ processors and $K = 10$ processes. The service time of the processors is exponentially distributed with mean value $1/\mu_1 = 76$ sec. The processes communicate with each other via finite buffers with uniform probability. The number of buffers is 10 and the capacity of the buffers is $M_i = 4$ for $i = 1, 2, \ldots, 10$. The total number of messages circulating in the system is $C = 35$. The mean communication time of the processes is given in Table 5.

From the process communication model we compute the following results for mean blocking times and blocking probabilities:

From the global model we obtain the performance measure values which are plotted in Figures 7a, b, c, d.

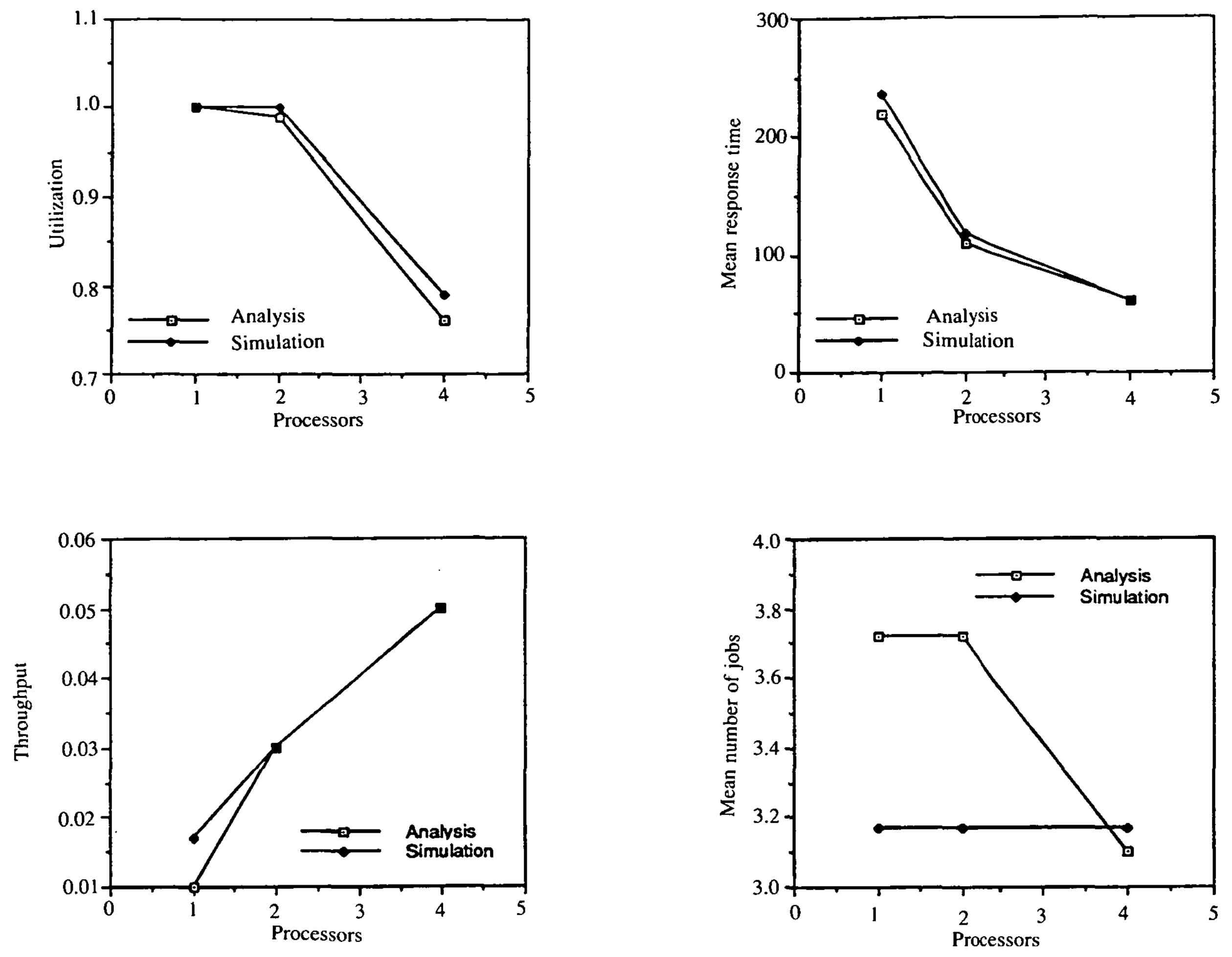

Figure 6. (a) Utilisation. (b) Mean response time. (c) Throughput, (d) Mean number of jobs.

Table 5

Process no.	$1/\mu$
1	77.52
2	5.56
3	10.31
4	9.04
5	19.82
6	9.01
7	13.89
8	13.79
9	13.72
10	13.89

Table 6

	Blocking Probabilities			Mean Blocking Times		
Process no.	An.	Sim.	Dev.	An.	Sim.	Dev.
1	0.465	0.492	5.900	7.461	7.722	3.481
2	0.695	0.713	2.600	8.232	8.511	3.214
3	0.592	0.624	5.222	15.791	16.062	1.723
4	0.001	0.004	0.000	0.000	4.431	0.000
5	0.764	0.746	2.434	29.211	27.911	4.442
6	0.528	0.546	3.311	9.521	10.663	11.911
7	0.179	0.184	3.112	11.261	11.962	6.211
8	0.184	0.199	8.410	11.982	12.351	3.101
9	0.521	0.501	3.822	10.453	9.673	7.501
10	0.523	0.498	4.611	13.250	12.831	3.201

6. CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

In this work we have analysed a multiprocessor system model where the processes communicate with each other via finite buffers using SEND and WAIT operations. The processes can block, if the communication operations (SEND and WAIT) cannot be executed. The system features are described in Section 2. The performance analysis of the model is carried out by developing two queueing network models which are analysed hierarchically. The inner model (so-called process communication model) is a closed queueing network model with blocking and captures the process communications in the multiprocessor system model. We have discussed two solutions for the process communication model. The first technique provides exact solutions for two-station queueing networks with blocking. The second technique, mean value analysis for blocking queueing networks provides approximate results for performance measures. From the process communication model we determined the blocking probabilities and blocking times which are used as input parameters for the outer model (so-called

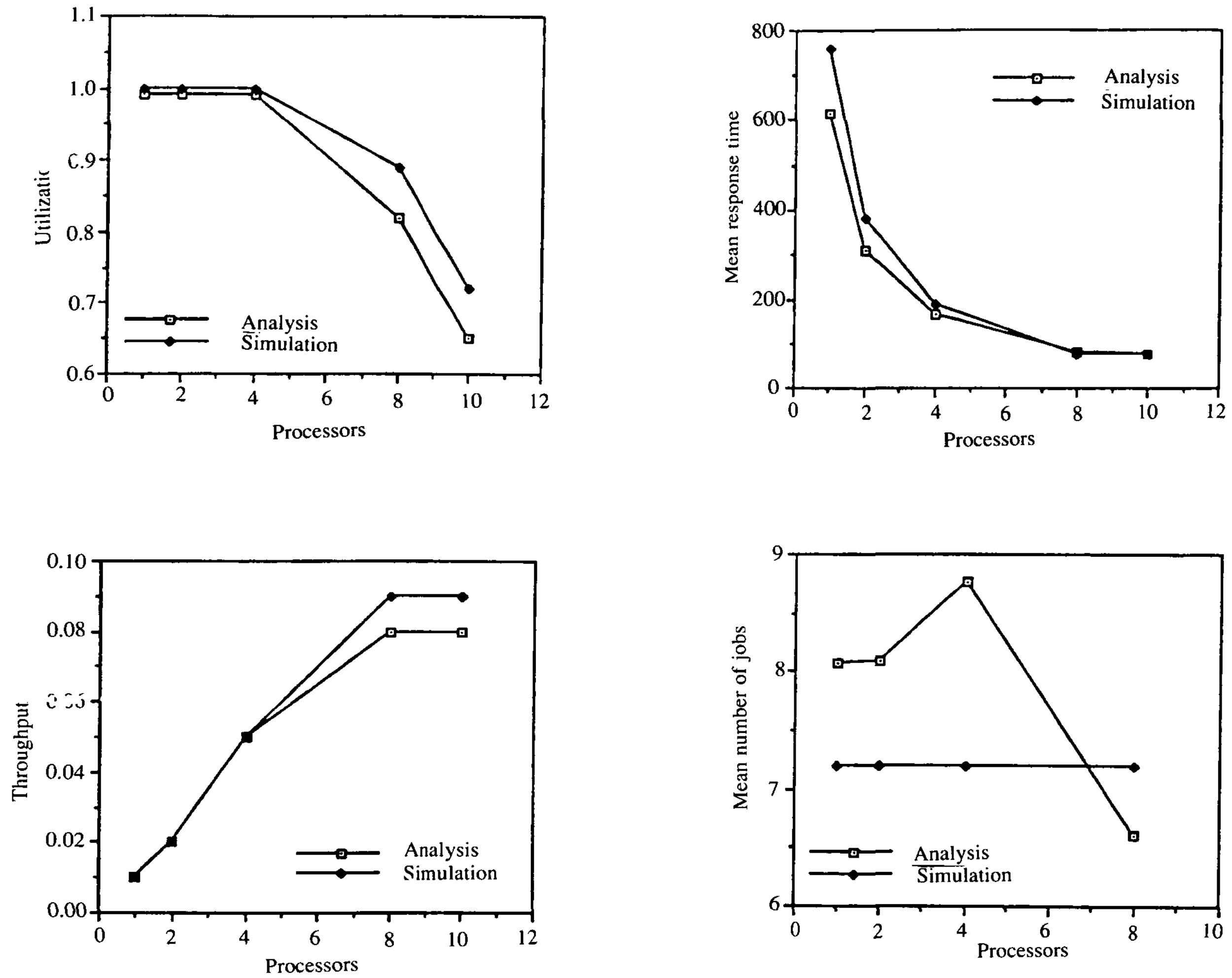

Figure 7. (a) Utilisation. (b) Mean response time, (c) Throughput. (d) Mean number of jobs.

global model) which is then analysed using product form algorithms.

In Section 5 we have shown three examples which were analysed by the suggested hierarchical concept. In all three examples we demonstrated that our analytical results are matching the simulation results. As demonstrated the deviations are between 1–17%. In all three examples we have seen that the performance measures show similar behaviour. The utilisation decreases with the increasing number of processors. This is due to the fact of the ratio of the total number of processes to the total number of processors. When the number of processes is less than the number of processors then it is clear that some of the processors will be idle. Consequently the utilisation of the processors will go down. We also observe that the throughput in all examples increases with the increasing number of processors. This makes sense because the more processors we have in the system the more processes can be serviced accordingly. The mean response time (duration of processes in the ready and active state) decreases with the increasing number of processors. This is an implication of the throughput behaviour that the processes are serviced in high productivity with the increasing number of processors. Another important aspect is that the more processors are in the system the more processes are involved in process communication which affects that the most of the processes are spending their times in the blocked state. For the mean number of processes the deviations between the approximate results and simulation are larger than in other cases. While in simulation results the mean number of processes in active and ready state are independent of the number of processors, the analytical results reveal the sensitivity of this measure towards the number of processors. Even though in the first example there is no significant change, however, in second and third examples the mean number of processes decreases significantly when more than 5 processors are in the system. This is the implication from throughput and response time that the processes will be serviced quickly and spending most of their times in the blocked state.

The suggested hierarchical analysis concept is valid for the processor sharing (PS) scheduling discipline, since all processes because of very small time slices can be active simultaneously. It would be interesting to analyse the FCFS case which we have investigated in the simulation study. There is a need for a new concept for the analytical investigation of the multiprocessor system in case of FCFS discipline. This work can also be extended to such cases where the messages are distinguished, i.e. a process can be blocked if it cannot find a specific message in its buffer. Another extension of this work could be reduction of the number of buffers, e.g. a common buffer for sent messages and another for receiving messages could be regarded in the system.

REFERENCES

1. I. F. Akyildiz, Exact product form solution for queueing networks with blocking, *IEEE Transactions on Computers*, **C-36**, (1), 122–125 (1987).
2. I. F. Akyildiz, Mean value analysis for blocking queueing networks, *IEEE Transactions on Software Engineering*, **SE-14** (4), 418–429 (1988).
3. I. F. Akyildiz and H. G. Perros, Special issue on queueing networks with finite capacity queues: Introduction, *Performance Evaluation* **10** (3) (1989).
4. F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios, Open, closed and mixed networks of queues with different classes of customers, *Journal of the ACM* **22**, **15**, 248–260 (1975).
5. A. Conway and N. Georganas, Queueing networks: Exact computational algorithms, *MIT Press* (1989).
6. K. Irani and I. H. Onyuksel, A closed form solution for the performance analysis of multiple bus multiprocessor systems, *IEEE Transactions on Computers* **C-33**, (11), 1004–1012 (1984).
7. S. Kundu and I. F. Akyildiz, Deadlock free buffer allocation in closed queueing networks, *Queueing Systems Journal: Theory and Applications*, **4**, 47–56 (1989).
8. T. Lang, M. Valero and I. Alegre, Bandwidth of crossbar and multiple bus connections for multiprocessors, *IEEE Transactions on Computers* **C-31**, 1227–1234 (1982).
9. M. A. Marsan and M. Gerla, Markov models for multiple bus multiprocessor systems, *IEEE Transactions on Computers* **C-31**, 239–248 (1982).
10. M. A. Marsan, G. Conte and G. Balbo, A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems, *ACM Transactions on Computer Systems* **2**, 93–122 (1984).
11. M. A. Marsan, G. Balbo and G. Conte, Performance models of multiprocessor systems, *The MIT Press* (1986).
12. M. Reiser and S. Lavenberg, Mean value analysis of closed multichain queueing networks, *Journal of the ACM* **27**, (2), 313–322 (1980).
13. D. Towsley, An approximate analysis of multiprocessor systems, *IEEE Transactions on Computers* **C-35**, (1986).

Announcements

9–12 JUNE 1992

IFIP WG8.4 Working Conference on 'The Portable Office: Microprocessor cards as elements of distributed offices', Chateau Laurier Hotel, Ottawa, Canada.

New technology offers new opportunities for new applications. Often, new technical features trigger unexpected new openings to theoretical development. The Micro-processor (or IC) card promises to offer all of these. Although it is physically very small, it is also a filing system of a potentially large content, constituting a truly 'portable' part of a distributed information system. The high-volume data-carrying capability might also be viewed as providing the network functionality required to connect many information systems together. Looking at it another way, the microprocessor allows such cards to add 'intelligent' characteristics to one's application.

To date, most approaches to the use of IC cards have been very specific, with problems being tackled on an ad hoc and strictly one-off basis. Clearly, the time has come to bring the new capabilities into a wider perspective. Talking about a 'portable office' should not be thought of as a part of the commercial promotion of some specific product, but as an important characteristic of office information systems.

This Working Conference intends to bring together system researchers, software designers and industrial developers, who need to integrate the IC card in new forms of office systems. A new generation of cards is emerging, which is not only capable of providing 'electronic money' services, but also of forming very general, multi-structurable components, that can relate to information in a rich variety of ways. The conference aims at exploring the options in these new application areas, and discussing the conceptual structures and dynamic aspects involved.

Topics

1. Office systems and smart elements
   - Office communication via cards
   - Cards as a component in human–machine interaction
   - Personal computing environments
   - Access-control, identification and authentication in the office
2. IC cards and the office environment
   - Methodological approaches with IC cards in distributed information systems
   - IC cards and networks
   - IC cards and databases
3. Technical requirements for office integration
   - Multi-application cards (fundamental problems, data sharing, etc.)
   - Operating systems for IC cards
   - Interfaces between IC cards and information systems
   - Standardization
4. Experimental projects and general applications
   - Corporate cards
   - Administration cards
   - Hospital, University and other special domains
5. Future devices with enhanced functionality
   - Concepts for combining other computerised functions with those currently provided on IC cards

The conference structure will permit meetings of task groups, the presentation of technical papers, posters and videos, the running of tutorials and the demonstration of working systems. The proceedings will be published by Elsevier Science Publishers (North-Holland) and will be available to participants at the conference.

Conference organizer

Professor George M. White, Computer Science Department, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5. E-mail: GMWSL@acadvm1.uottawa.ca. Fax: (613) 564-9486.

For more information call

Kathy Mahoney, Conference Registrar, 340 March Road, Suite 400, Kanata, Ontario, Canada K2K 2E4. Tel: (613) 592-8160. Fax: (613) 592-8163.

27–29 JULY 1992

The 1992 Factory Automation Conference, Factory 2000, University of York, UK

'Factory' is being organised by the Institution of Electrical Engineers (IEE). This third international conference will consider the application and effects of modern technology, management and optimisation techniques in manufacturing industries.

The intended audience will include manufacturers who appreciate the need to respond to the rapidly changing market place and the new technologies being implemented by their competitors. It will particularly appeal to those who want to ensure that the most cost-effective technology is used and that there is a degree of fit between the manufacturing, management and marketing strategies.

For further information contact:
IEE Conference Services, Savoy Place, London WC2R 0BL, Tel: 071 240 1871, ext. 222.