On the Modeling of Shared Resources by Queueing Networks with Serialization Delays *

Anna Hać **

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley

ABSTRACT

This paper presents a new approach to estimating serialization delays in computer systems. Two types of models are introduced to describe synchronization of accesses to shared resources using simulation and numerical methods, respectively. The model parameters are estimated from the values of measurable quantities. For the analytic models, the probabilities that a shared resource is locked, thereby causing other processes to wait for the resource to be accessible, are calculated, under certain assumptions, as functions of the probabilities of access and of the number of active transactions. The service times of lock servers are calculated as functions of the loads on all service centers. Performance measures applicable to the evaluation of computer systems are introduced and evaluated. An example involving a distributed file system and measurement data collected in a small business installation is given to compare performance measures provided by the simulation and analytic models.

1. Introduction

The problem of exclusive access to shared resources was solved using critical sections that are accessed by controlling the values of the synchronization primitives (e.g., semaphores) [4]. In [5] it is shown that a queueing network model can be constructed to represent spin lock behavior. That model may be used to predict the performance degradation due to locking. The results of extensive simulations of systems with static and dynamic locking to study the effects of the granularity of locks in a database system are presented in [8, 9].

The approximate analysis of a complex database system presented in [3] permits us to model a database that can be accessed and modified concurrently by a number of users (i.e., transactions). That approach treats individual transactions in isolation and analyzes them in steady state. The influence of the other transactions is represented by the probability F that an entity (i.e., a resource) required by the transaction is unavailable, and by the probability F_1 that an entity which was unavailable at the time of the previous attempt to acquire it is still unavailable at the next attempt. However, the probability F that an attempt by a running transaction to get an entity will fail is assumed to be constant and independent of the number of entities held and of past history. The probability F_1 that an attempt by a waiting transaction to get an entity will fail is assumed to be independent of the number of entities held and of past history. These assumptions simplify the analysis but may be unacceptable in practice. The accuracy of the approximation is evaluated by simulation. However, only in 5 cases out of 12 examples presented

* The research reported here has been supported in part by the State of California and the NCR Corporation under MICRO Grant No. 1-532436-19900.

** On leave from Institute of Computer Science, Technical University of Warsaw, Poland.

the results are acceptably accurate. In the remaining 7 cases there are large differences between the results obtained from the analytical and simulation models. Moreover, the curves representing response time for the simulation model have shapes different from those of the curves produced by the mathematical model.

One of the known methods for evaluating computer systems and networks with serialization delays considers execution graphs [11]. The critical section delays are modeled by a wait service center with iteratively set service time. The model uses class changes to allow different service times in different serialization phases. The model can only be used to analyze a single class network.

Another method for modeling serialization delays is presented in [2] in the context of a software module modeling problem. These models are based on a two level description, the software level and the hardware level, of a computer system. The servers in a queueing network model are used to represent software modules which may be either reentrant or nonreentrant (i.e., serialized). However, the solution is limited to the case where all processing that occurs in a software module has to take place on a single processor, a single I/O subsystem and a number of terminals.

The effect of a two-phase locking scheme with a dynamic lock request policy using pseudo-servers to represent serialization delays is presented in [12]. The proposed solution scheme is iterative, since the loadings at the Lock-Wait stations are functions of the mean transaction response time. Some further results are presented in [13] for the case of a Markov chain model and validated using simulation. In [14] the evaluation of performance of centralized database systems with static locking (i.e., a scheme in which database transactions should acquire all locks before their activation) is presented. The computer system is modeled as a queueing network. Two scheduling policies are considered for the activation of a transaction. The scheduling overhead for scanning the blocked transactions is calculated. The number of transactions to be scanned is limited by a window size parameter. The model is analyzed using a hierarchical decomposition method and the results of the approximate solution are validated using simulation. Also investigated are the effects on performance of varying system parameters such as transaction size, granularity of locking, window size, the scheduling discipline, and their overhead.

Yet another approach is presented in [1], where serialization delays are represented by load dependent aggregate servers, with one additional server included for each source of serialization delay. The service times of the aggregate servers are iteratively determined by successive solutions of the model. However, the algorithm given in [1] only applies to single class closed queueing networks with load independent servers employing the processor sharing scheduling discipline.

A class marking technique approach is presented in [7], where the network is reduced to an equivalent network with a population constrained subsystem. A collection of multiple server service centers are used to represent population constraints. Each constraint is represented by a different service center with a number of servers equal to the maximum number of processes allowed in the subsystem. The service time of a server is iteratively refined by estimating the residence time of a user in the population constrained subsystem.

Most of the approaches briefly reviewed so far in this section are based on iterative algorithms to estimate serialization delays in computer systems and networks. These algorithms start by estimating the service time degradation due to locking; the queueing network is then solved, the service time is computed again and compared with that obtained in the previous step. If the estimates are sufficiently close, then the iteration is finished, otherwise the network is solved again with the new estimate replacing the old.

The approaches reviewed above do not consider the measurability of the model parameters, and some of the parameters they introduce are not easilv measurable, for instance the probabilities of entering the critical sections in [1] and [7].

This paper describes a practical and non-iterative method for evaluating queueing networks in which serialization delays are represented using measurable parameters. The probabilities that shared resources be locked, thereby causing other processes to wait for the resource to be accessible, are calculated, under certain assumptions, as functions of the probabilities of accessing the resource and of the number of active transactions. The service times for locks are calculated as functions of the loads on all service centers.

The model is applicable to the case of multiple classes and to that of a multiprocessor or multicomputer system.

As an example, the case of a distributed file system will be discussed to illustrate our method for modeling access to the shared resources. Simulation and analytic models will be constructed for a file disk to represent the serialization delays which may arise when a file is read or written. Further results for a distributed file system with dynamic locking will be presented in [6].

The values of performance measures obtained from the two types of models will be compared using measurement data collected in a small business installation.

2. Modeling Locks by Simulation

A simulation model was constructed for a file disk accessed concurrently by transactions which can read or write it. A diagram representing the model is shown in Fig. 1. The policy for accessing a file is as follows:

(a) a transaction can read a file iff it is not being written;

(b) a transaction can write a file iff it is not being read or written.

The policy for servicing the requests is FCFS (i.e., first come first served).

The model is implemented using the RESQ2 queueing network analysis package [10] running on an IBM 4341 (VM/370) system. The RESQ2 package provides a method for representing exclusive access to shared resources using passive queues with an arbitrary number of nodes. There are allocate, release, create and destroy nodes. Each passive queue has its pool of tokens. The nodes access the tokens in the pool.

In the simulation model of the file disk, three passive queues are used to synchronize access to a file. The initial number of tokens in the pool of the first passive queue is equal to 1; a token is allocated when either a write access or a read access is made to a file; in the case of write access, the token is released when the file has been written; in the case of read access the token is released immediately after a token in the pool of the second passive queue has been allocated. In the case of a read access, the first passive queue is used to grant exclusive access to the second passive queue. The initial number of tokens in the pool of the second passive queue is equal to the highest integer in the implementation, and tokens are allocated in the case of read access to a file and released when a file has been read. The initial number of tokens in the pool of the third passive queue is equal to zero and a token is created after completion of every read access to the file. These tokens are allocated and immediately destroyed at the beginning of any write access to the file (to perform a write access, the number of tokens in the pool of the second passive queue must be equal to the highest integer, as this means that all reads, which had required access to the file before the arrival of a write request, have been completed); if a write initiates access to the file, then the number of tokens in the pool of the second passive queue is equal to the highest integer and the write access can proceed. When a write and a read require access to the file at the same time, then it is uncertain which one should proceed first: it appears a race to allocate a token from the pool of the first passive queue.

With respect to file access, a process can be in any of the following states: running, waiting, committing, and aborting. The running state represents a period of execution in a file disk. A process is in the waiting state while it awaits in a node for a token to be allocated. The committing state indicates a successful completion of a process: all tokens are released. The aborting state follows an unsuccessful attempt to obtain that the number of tokens in the pool of the second passive queue become equal to the highest integer. A token can be allocated, released, created and destroyed in a node. A state/node diagram for a process is shown in Fig. 2.

3. Modeling Locks by a Numerical Method

An analytic model was constructed for the same file disk accessed concurrently by transactions which can read or write it. The diagram of a file disk with service centers representing serialization delays for reading and writing a file is shown in Fig. 3. The disk server represents a critical section in which a file is read or written. The transactions waiting in the queue to this server have already acquired their locks. The read lock server and its queue represent the delay experienced by transactions which require read access when the file is being written. The mean service time of this server is estimated by the mean remaining time spent in the critical section by a transaction which performs write access, assuming that the lock is released immediately when the file has been written. The write lock server and its queue represent the delay experienced by transactions which require write access when the file is being read or written. The mean service time of this server is estimated by the mean remaining time spent in the critical section by a transaction which performs read or write access, assuming that the lock is released immediately when the file has been read or written. The direct path models the case in which the file is found to be unlocked. The probabilities of accessing lock servers and their service times are calculated below.

Let PR and PW be the relative probabilities of reading and writing a file (i.e. $PW+PR=pf$, where pf is the probability of accessing that file), with

$$PR=pr_{file}*pf, \tag{1}$$

$$PW=pw_{file}*pf, \tag{2}$$

where pr_{file} and pw_{file} are the unconditional probabilities of reading and writing a file, respectively, and

$$pr_{file}+pw_{file}=1.$$

Let n be the number of active transactions within a host, and nw be the average number of active transactions which the next access to the file is the write access; then $n-nw$ is the average number of active transactions which the next access to the file is the read access. The probabilities of a read request (pfr) and of a write request (pfw) for a file that happens to be locked are calculated as follows:

$$pfr=1\text{-}(1\text{-}PW)^{nw}, \tag{3}$$

$$pfw=1\text{-}(1\text{-}PR\text{-}PW)^{n-1}. \tag{4}$$

This means that

(1) a transaction requesting to read a file is delayed iff one transaction is writing the file, and

(2) a transaction requesting to write a file is delayed iff at least one transaction is reading or writing the file.

The probabilities of serialization delays are calculated on the basis of the relationships:

$$P(R\cup W)=P(R)+P(W)\text{-}P(R\cap W)=P(R)+P(W)\text{-}P(R)P(W), \tag{5}$$

$$P(\overline{R\cup W})=P(\overline{R}\cap\overline{W})=(1\text{-}P(R))(1\text{-}P(W)), \tag{6}$$

where $\overline{A}$ is the complement of A.

These relationships are based on the assumption that the probability $P(R)$ that a read lock is set and the probability $P(W)$ that a write lock is set, are independent (i.e., the events R and W are mutually exclusive).

Hence, considering (3) and (4), the probabilities of nonzero serialization delays for reading $(P(R))$ and, respectively, writing $(P(W))$ a file are given by

$$P(R)=pfr-pfr*pfw\frac{pfr}{pfr+pfw}, \tag{7}$$

$$P(W)=pfw-pfr*pfw\frac{pfw}{pfr+pfw}, \tag{8}$$

where $\frac{pfr}{pfr+pfw}$ and $\frac{pfw}{pfr+pfw}$ are the weighting factors for $pfr*pfw$.

The probability that there is no serialization delay is given by

$$P(\overline{R \cup W})=(1-pfr)(1-pfw). \tag{9}$$

Let s_{read} and s_{write} be the mean times during which the first transaction in the queue to the disk has to wait when a file is being read or written, respectively. These times are given by

$$s_{read}=min(max((n-nw)(pf*s_{file}-\sum_{i=1}^{M-1} p_i s_i-\frac{tth}{n}), pf\frac{s_{file}(n-nw)}{n*nw}), s_{file}), \tag{10}$$

$$s_{write}=min(max(nw(pf*s_{file}-\sum_{i=1}^{M-1} p_i s_i-\frac{tth}{n}), pf\frac{s_{file}nw}{n(n-nw)}), s_{file}), \tag{11}$$

where M is the number of service centers in the system, the s_i's are the mean service times of the service centers, and *tth* is the mean "think" time for the active transactions.

The mean service times of servers representing serialization delays due to the reading and, respectively, writing of a file are given by

$$s_{read}^{lock}=s_{write}, \tag{12}$$

$$s_{write}^{lock}=min(s_{read}+s_{write}, s_{file}). \tag{13}$$

These times are limited by the value of the mean service time in the disk, since the lock is released immediately when the file has been read or written.

The diagram of a file disk shown in Fig. 3 represents the case in which the disk has only one file on it. In the model of a file disk with many files on it, the portion containing read and write lock servers is repeated for every file, but there is still only one disk server. In this case, the probabilities and service times for lock servers are calculated in the same way; however, n becomes the number of active transactions accessing a given file, and may be different for different files.

4. Examples of Model Applications

Simulation and numerical methods were applied to solving the following two types of models:

(a) a model of a centralized system;

(b) a model of a distributed system containing several identical hosts.

The model of each host represents a number of terminals, display outputs, a CPU, a directory server and a number of local and global files (in the centralized system there are only local files). The model of a single host and that of the multiple host system are shown in Fig. 4 and 5, respectively. It is assumed that each host has a directory located on a separate disk. The directory is modeled in the same way as a file. The CPU overhead due to disk accesses is incorporated into the CPU service time.

The workload of the system consists of a number of transactions. It is assumed that each transaction type is associated with a terminal. It is also assumed that each type of transaction defines its own class (i.e., the number of classes is equal to the number of transaction types). In our distributed system model one extra class is added for each host to represent all types of transactions from the other hosts accessing it. A class is characterized by the following measurable parameters:

(a) the number of transactions (i.e., the number of terminals of this class);

(b) the mean value of "think" time of each interaction in each transaction type;

(c) the mean value of the display output time;

(d) the mean service times in all service centers (the service time in a directory or file service center is equal to the time of physical disk access plus transmission time);

(e) the probabilities of accessing each service center (including the file disks) and each host.

In the model of a distributed system the class describing transactions from remote hosts accessing the local host is characterized by the following measurable parameters:

(a) the number of transactions (i.e., the number of terminals of each class in each remote host weighted by the probability of accessing the local host from the remote host for each class);

(b) the mean service times in all service centers;

(c) the probabilities of accessing each service center (including the file disks) and each host.

The following assumptions are made:

(1) a transaction does not change its class within a host;

(2) all transactions from the remote hosts create a separate class in the local host; these transactions are statistically identical in the local host and, after leaving the local host, the returning transactions are distributed to their original sites probabilistically rather than deterministically, as will be shown later in this section, (this assumption is made to simplify the model since a transaction accesses a global file infrequently; otherwise, each transaction type from a remote host should be represented by a separate class in the local host);

(3) the probabilities of reading and writing are associated with files (this assumption is needed only for practical reasons, to separate files which are mostly read from those that are mostly written, since the read and write accesses cause different degradations of performance).

The behavior of a transaction is as follows: each transaction has to be processed in the CPU before it accesses any other service center; it may also be processed in the directory server and in a file disk; it may be delayed because of locks and be processed by the network server in the distributed system model. In its local host a transaction also produces several display outputs before returning to the user terminal.

Because of the lock servers, the directory and file disk are modeled as the FCFS servers, the classes of transactions are not distinguished in these servers. The probabilities of accessing the CPU from the file disks are calculated below, assuming that the classes of transactions in the other service centers are distinguished. Let nc be the number of classes of transactions within a host, pf_{ij} be the probability of accessing file j by a transaction of class i, pdr_i be the probability of accessing the directory server by a transaction of class i, and n_i be the number of active transactions of class i. The probability of accessing the CPU after having accessed file j for every transaction of class i is given by

$$p_{ij}^{cpu} = \frac{pf_{ij}\, n_i}{\sum_{i=1}^{nc} pf_{ij}\, n_i}, \tag{14}$$

and, after having accessed the directory server, is given by

$$p_i^{cpu} = \frac{pdr_i\, n_i}{\sum_{i=1}^{nc} pdr_i\, n_i}. \tag{15}$$

In a similar way, the transactions are directed to their destination when leaving a remote host to return to their local hosts. Let pn_{jki} be the probability of accessing host j by a transaction of class i on host k, and n_{ki} be the number of active transactions of class i in host k. The number of active transactions accessing host j from host k is given by

$$n_{jk} = \sum_{i=1}^{nc} pn_{jki}\, n_{ki}. \tag{16}$$

Let ph_j be the probability that host j be exited by remote transactions, and NH be the number of hosts. The probability of accessing host k is given by

$$p_k^{host} = \frac{n_{jk}\, ph_j}{\sum_{m=1}^{NH} n_{jm}\, ph_j}. \tag{17}$$

The simulation and analytic models were constructed for both centralized and distributed systems. The results obtained from the simulation model with different types of transactions are compared in Section 5 with those from the analytic model.

The following parameters can be estimated by monitoring a real installation:

(1) the mean service time of each service center in the network (the service time in a directory or file service center is equal to the physical disk access time plus the transmission time);

(2) the probabilities of accessing each service center (including file disks) for each class in the model;

(3) the probabilities of accessing remote hosts from a local host;

(4) the unconditional probabilities of reading and writing each file;

(5) the number of users (i.e., of active transactions);

(6) the average number of users (active transactions) requiring write access to a file.

The parameters just listed are needed to calculate both the probabilities that shared data be locked and the mean service times for locks.

The performance measures produced by solving the models described above are:

(1) the utilization of the service centers,

(2) the system throughput,

(3) the mean system response time.

These performance measures are calculated under the condition that the model parameters be measurable.

5. Some Results

Two simple examples of performance prediction in the centralized and distributed systems using measurements collected in a small business installation show the application of the models and the performance degradation due to locking.

The workload of the system is defined by five types of transactions. The following data are measured for each transaction type:

I - the number of interactions;
T_{TH} - the mean "think" time of an interaction;
T_S - the mean time of a display output;
T_{CPU} - the mean CPU time consumed by a transaction of that type;
D - the total number of disk I/O operations for the files (D_F) and the directory server (D_D) in a transaction of that type;
S - the total number of display outputs in a transaction of that type;
F - the number of files;
P - the percentage of the number of transactions of that type in the workload.

The following parameter values can be derived from the measurement data:

Average CPU time per interaction $(\frac{T_{CPU}}{I})$

Average number of disk I/O operations per interaction $(\frac{D}{I})$

Average number of display outputs per interaction $(\frac{S}{I})$

Average CPU time between successive disk I/O operations and/or display outputs $(\frac{T_{CPU}}{D+S})$

Probability of accessing each file disk $(\frac{D_F}{1+D+S})$

Probability of accessing the directory server $(\frac{D_D}{1+D+S})$

Probability of a display output $(\frac{S}{1+D+S})$

The unconditional probabilities of reading and writing files and the number of writing accesses made by the active transactions can be estimated from the number of read and write accesses made by each transaction.

The estimated values of the parameters of each transaction type (i.e., each class in the model) are given in Tables I and II for the single host system and for a system with two hosts, respectively. In Table II only the values of parameters different from those in Table I are shown. Also, the probability of accessing the remote host (P_H) is derived assuming that 10 percent of the disk I/O operations (equal to H) for each transaction type are performed in the remote host. The probabilities of accessing a file disk (P_F) and directory server (P_D) by remote transactions are calculated based on the number of local accesses. Note that the remote transactions do not make display outputs and do not return to a user terminal in the remote host.

Table I. Estimated values of the model parameters for a single host system.

Transaction Type	$T_{CPU}/(D+S)$ [msec]	T_{TH} [msec]	T_S [msec]	$D_F/(1+D+S)$	$D_D/(1+D+S)$	$S/(1+D+S)$
1	42	1000	15	0.4	0.03	0.47
2	35	10000	25	0.75	0.05	0.18
3	224	1000	20	0.7	0.05	0.19
4	32	500	15	0.9	0.066	0.025
5	15	2000	15	0.3	0.02	0.6

Table II. Estimated values of the model parameters for a system of two hosts.

Transaction Type	$D_F/(1+D+S+H)$	$D_D/(1+D+S+H)$	$S/(1+D+S+H)$	P_H	P_F	P_D
1	0.38	0.03	0.46	0.04	0.75	0.06
2	0.7	0.05	0.17	0.07	0.92	0.07
3	0.67	0.05	0.18	0.07	0.88	0.06
4	0.83	0.06	0.023	0.08	0.926	0.067
5	0.3	0.02	0.6	0.03	0.74	0.05

The average service times for the following two service centers are the same for all transaction types:

Disk access	$T_{DISK}=30msec$
Network server	$T_{NET}=0.2msec$

In the first example, the performance measures are computed for a centralized system containing one directory server and one file disk. The system was modeled using simulation and numerical methods. The confidence level of the simulation was 90 percent. The confidence interval widths was 30 percent. Performance measures were calculated for the simulation and analytic models including serialization delays and, separately, for the analytic model not including servers representing serialization delays, to determine the degradation of performance due to locking. The results of the comparison are presented in Tables III and IV.

Table III. Performance measures for the simulation and analytic models with serialization delays.

Model*	Transaction Types	Number of Transactions of Each Class	Performance Measures				
			Utilization			Throughput [1/sec]	Response Time [sec]
			CPU	File	Directory		
SLOCK	1; 2	1; 1	0.37	0.159	0.01	0.63	1.05
ALOCK	1; 2	1; 1	0.34	0.152	0.01	0.53	1.28
SLOCK	1; 2	2; 2	0.64	0.29	0.02	0.88	1.55
ALOCK	1; 2	2; 2	0.62	0.28	0.02	0.97	1.63
SLOCK	1; 2	4; 4	0.94	0.42	0.03	1.55	2.69
ALOCK	1; 2	4; 4	0.93	0.42	0.03	1.45	3.00

* SLOCK - simulation, ALOCK - analytic

Table IV. Performance measures for the analytic models (with and without serialization delays).

Model*	Transaction Types	Number of Transactions of Each Class	Performance Measures				
			Utilization			Throughput [1/sec]	Response Time [sec]
			CPU	File	Directory		
ALOCK	1; 2	1; 1	0.34	0.152	0.01	0.53	1.28
NLOCK	1; 2	1; 1	0.35	0.156	0.01	0.54	1.19
ALOCK	1; 2	2; 2	0.62	0.28	0.02	0.97	1.63
NLOCK	1; 2	2; 2	0.63	0.28	0.02	0.98	1.57
ALOCK	1; 2	4; 4	0.93	0.42	0.03	1.45	3.00
NLOCK	1; 2	4; 4	0.94	0.42	0.03	1.46	2.99

* ALOCK - analytic, NLOCK - analytic without locks

In the second example, the performance measures are computed for a distributed system containing two identical hosts. Each host includes one directory server and two file disks. Two analytic models were solved: one including serialization delays, the other one not including them, to determine the degradation of performance due to locking. A simulation model was not run since, as shown in Table III, serialization delays can be modeled analytically with reasonable accuracy. The results of the comparison are presented in Table V.

Table V. Performance measures for the distributed system model.

Model*	Host Number	Transaction Types	Number of Transactions of Each Class	Performance Measures					
				Utilization				Throughput [1/sec]	Response Time [sec]
				CPU	File(1)	File(2)	Directory		
ALOCK	1	1; 2	2; 2	0.62	0.14	0.14	0.019	0.46	8.25
	2	4; 5	2; 2	0.39	0.14	0.14	0.019	0.46	15.4
NLOCK	1	1; 2	2; 2	0.63	0.14	0.14	0.019	0.47	7.94
	2	4; 5	2; 2	0.40	0.14	0.14	0.019	0.47	15.1
ALOCK	1	1; 2	4; 4	0.95	0.21	0.21	0.029	0.71	13.6
	2	4; 5	4; 4	0.60	0.21	0.21	0.029	0.71	20.8
NLOCK	1	1; 2	4; 4	0.95	0.21	0.21	0.029	0.71	13.7
	2	4; 5	4; 4	0.59	0.21	0.21	0.029	0.71	20.9
ALOCK	1	1; 2	8; 8	1.00	0.22	0.22	0.031	0.75	33.9
	2	4; 5	8; 8	0.63	0.22	0.22	0.031	0.75	41.1
NLOCK	1	1; 2	8; 8	1.00	0.22	0.22	0.031	0.75	33.9
	2	4; 5	8; 8	0.63	0.22	0.22	0.031	0.75	41.1

* ALOCK - analytic, NLOCK - analytic without locks

The comparison of the results from the models shows that the performance degradation due to locking is negligible. Other simulation and numerical experiments were performed, also for non-CPU-bound systems, but the overhead due to locking was always found to be very small. Indeed, the lock service times are relatively small with respect to the service times of the other service centers. Moreover, the lock service times are limited by the service time of the corresponding file disk (see equations (10)-(13)). That is why the servers representing serialization delays can never become the system bottlenecks, and, even when the system is not saturated, their effect on system performance is negligible. If the system is saturated and a disk becomes a system bottleneck, then the servers representing serialization delays can be highly utilized but their utilization is always smaller than the utilization of the disk. In that case the time delays due to locking can be significant, but they have negligible effect on performance, since the disk server is saturated.

We also performed experiments with various lock granularities. Locks were assumed to be on records, sectors, or files. The values of the performance measures were very slightly influenced by the serialization delays due to locking.

An example of a centralized system model with not derived from measurements data is presented to show the effect of serialization delays on performance when file disk utilization is high. There is one class of users with 20 msec "think" time, 30 msec CPU time, 60 msec disk access time, probabilities 0.9 and 0.04 of accessing the file disk and the directory, respectively, probability 0.5 of reading or writing the file or the directory, and no display outputs. The results of simulation and analytic models are presented in Table VI.

Table VI. Performance measures for the analytic models (with and without serialization delays).

Model*	Number of Transactions	Performance Measures				
		Utilization			Throughput	Response Time
		CPU	File	Directory	[1/sec]	[sec]
ALOCK	4	0.47	0.85	0.04	0.95	4.19
NLOCK	4	0.53	0.95	0.04	1.06	3.76
ALOCK	8	0.54	0.97	0.05	1.08	7.39
NLOCK	8	0.55	0.99	0.04	1.11	7.21
ALOCK	12	0.55	0.99	0.04	1.11	10.82
NLOCK	12	0.55	0.99	0.04	1.11	10.78

* ALOCK - analytic, NLOCK - analytic without locks

6. Conclusion

The approach presented in this paper permits the evaluation of queueing networks with serialization delays. Simulation using passive queues is an accurate method for modeling real systems. The calculation of the probabilities and the service times of lock servers for an analytic model is based on the analysis of the accesses to the resource (in our examples, of the read and write accesses to a file), and is independent of performance measures. The comparison of the results obtained from simulation and from numerical techniques showed that the locking overhead is usually negligible, since the service times of lock servers tend to be relatively small in comparison with the service times of the other service centers. The mean service times of lock servers are limited by the mean service time in the disk. This is why the servers representing serialization delays are never system bottlenecks, and, even when the system is not saturated, their effect on the system performance is generally negligible. Also, changes in the granularity of locking do not seem to have any significant impact on performance, even though the time delays due to locking in a file disk is significant. The reason of these results is that the lock is released immediately when the file has been read or written.

The more general case for the distributed file system with dynamic locking, when a file may be locked by a transaction which is then sent to the other service centers without releasing the lock, will be shown in [6]. In this case the performance degradation due to locking can be significant; also, the changes in the granularity of locking have impact on the system's performance.

7. Acknowledgments

The author wishes to thank Domenico Ferrari for his encouragement and support during the preparation of the paper and for his helpful comments on the manuscript. She also would like to thank T. Paul Lee for many helpful discussions on the subject of the paper and for his suggestions in preparing the figures used in this paper. Thanks are also due to the Research Division of the IBM Corporation, whose queueing network analysis package RESQ2 played a very useful role in the research reported in this paper.

8. References

[1] S. C. Agrawal and J. P. Buzen, The Aggregate Server Method for Analyzing Serialization Delays in Computer Systems, Performance Evaluation Review: Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (Aug.-Sept. 1982).

[2] J. R. Agre and S. K. Tripathi, Modeling Reentrant and Nonreentrant Software, Performance Evaluation Review: Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (Aug.-Sept. 1982) 163-178.

[3] A. Chesnais, E. Gelenbe and I. Mitrani, On the Modeling of Parallel Access to Shared Data, Comm. ACM, 26, 3 (1983) 196-202.

[4] E. W. Dijkstra, The Structure of THE Multiprogramming System, Comm. ACM, 11, 5 (1968) 341-346.

[5] D. C. Gilbert, Modeling Spin Locks with Queueing Networks, ACM Operating Syst. Review, 12, 1 (1978) 29-42.

[6] A. Hać, A Decomposition Solution to a Queueing Network Model of a Distributed File System with Dynamic Locking, in preparation.

[7] P. A. Jacobson and E. D. Lazowska, A Reduction Technique for Evaluating Queueing Networks with Serialization Delays, Proc. Performance'83 (1983) 45-59.

[8] D. R. Ries and M. R. Stonebraker, Effects of locking granularity in a database management system, ACM Trans. Database Systems, 2, 3 (1977) 233-246.

[9] D. R. Ries and M. R. Stonebraker, Locking granularity revisited, ACM Trans. Database Systems, 4, 2 (1979) 210-227.

[10] C. H. Sauer, E. A. MacNair and J. F. Kurose, The Research Queueing Package, Version 2: CMS Users Guide, IBM Research Report RA 139, 41127 (April 1982).

[11] C. Smith and J. C. Browne, Aspects of Software Design Analysis: Concurrency and Blocking, Proc. Performance'80 (1980) 245-253.

[12] A. Thomasian, An Iterative Solution to the Queueing Network Model of a DBMS with Dynamic Locking, Proc. 13 Comp. Measurement Group Int'l Conference, San Diego, CA (Dec. 14-17, 1982) 252-261.

[13] A. Thomasian, Queueing Network Models to Estimate Serialization Delays in Computer Systems, Proc. Performance'83 (1983).

[14] A. Thomasian and I. K. Ryu, A Decomposition Solution to the Queueing Network Model of the Centralized DBMS with Static Locking, Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (Aug. 29-31, 1983) 82-92.

nodes:

△ allocate

▽ release

⧍ create

⧩ destroy

◇ obtains the number of tokens in the second pool and if it is the highest integer (2147483647), then the YES way is chosen, otherwise the NO way is chosen

number of tokens in the pools:

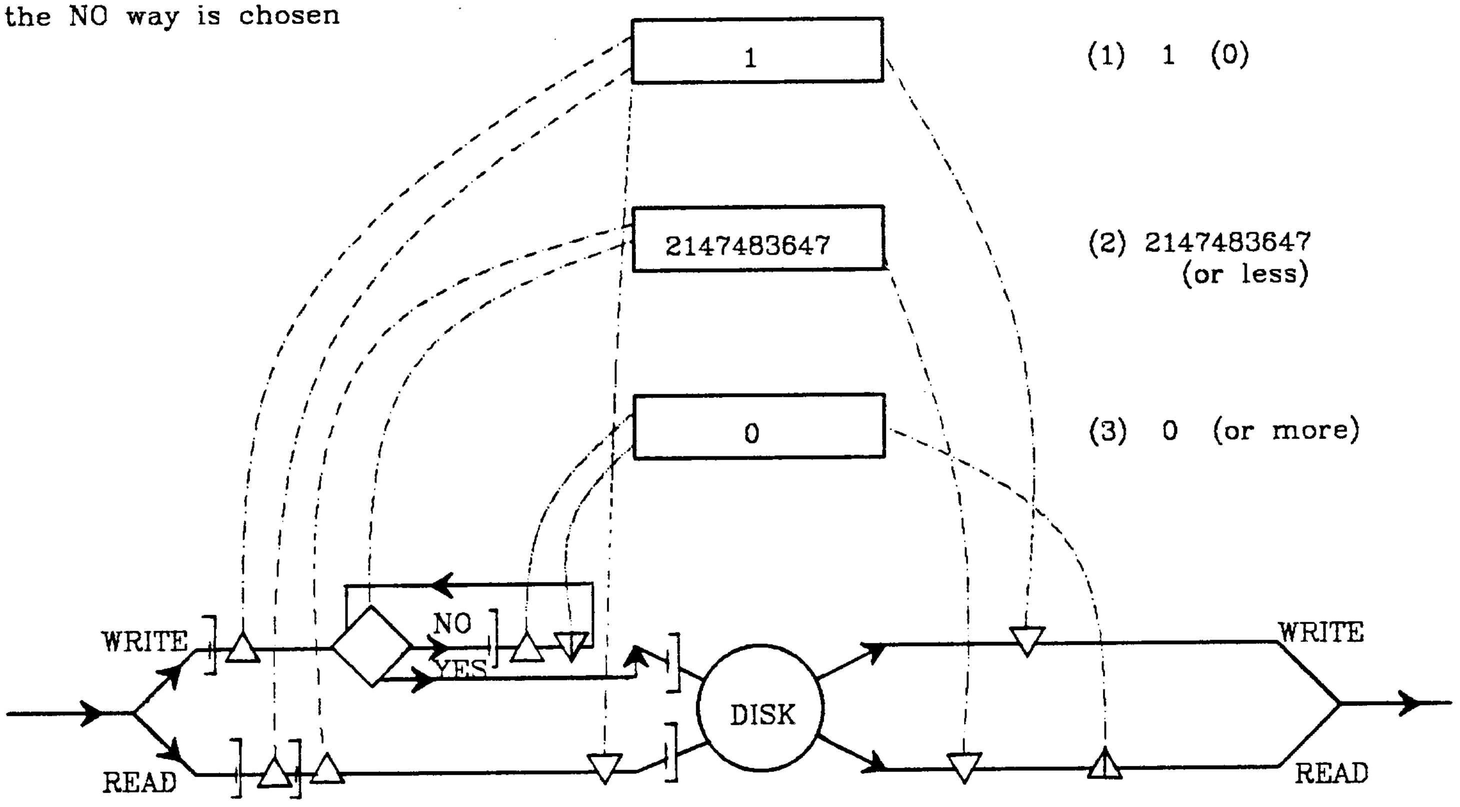


Fig. 1. The simulation model of a file disk.

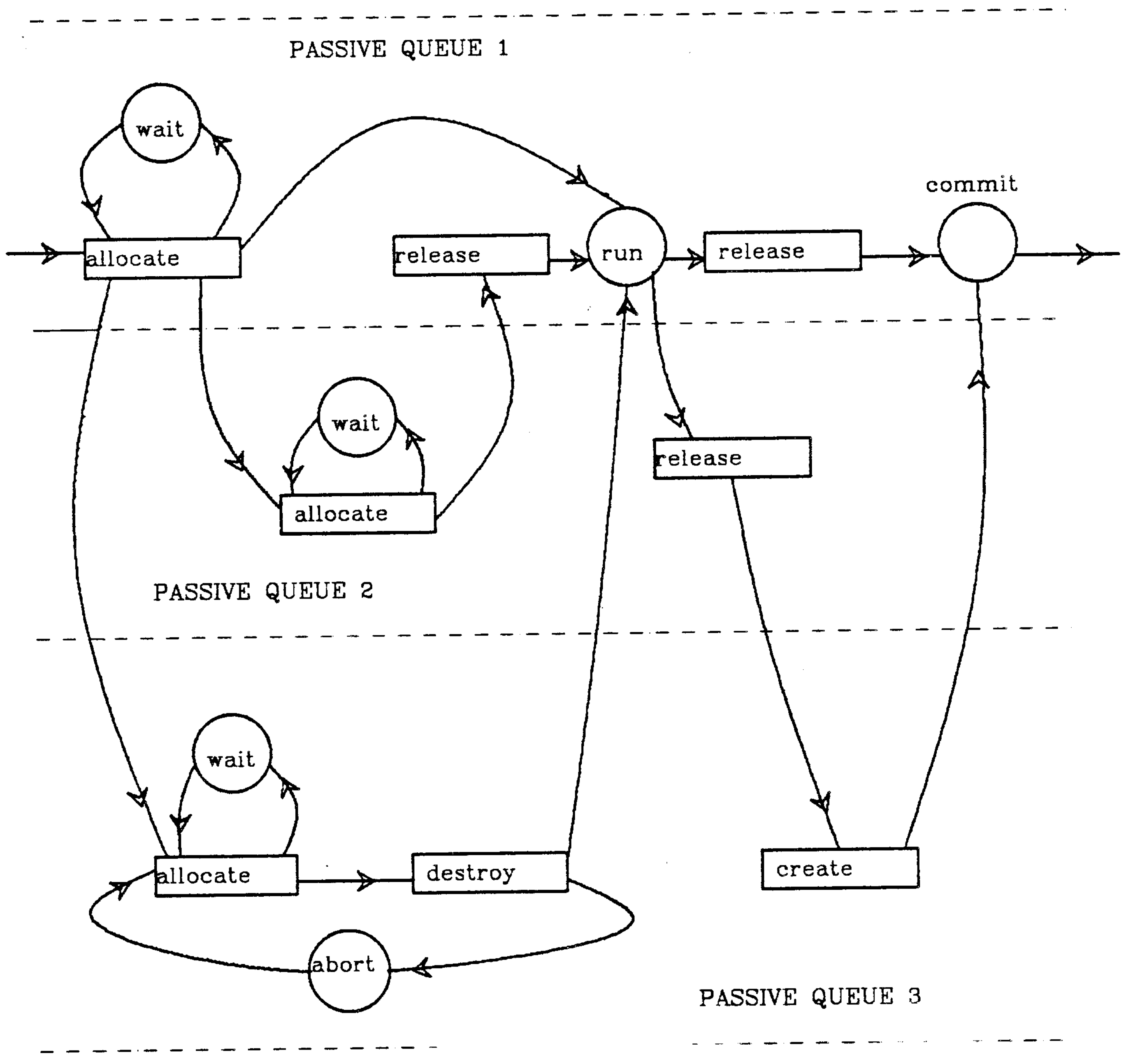
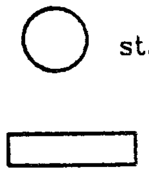


Fig. 2. The state/node diagram

for a process.



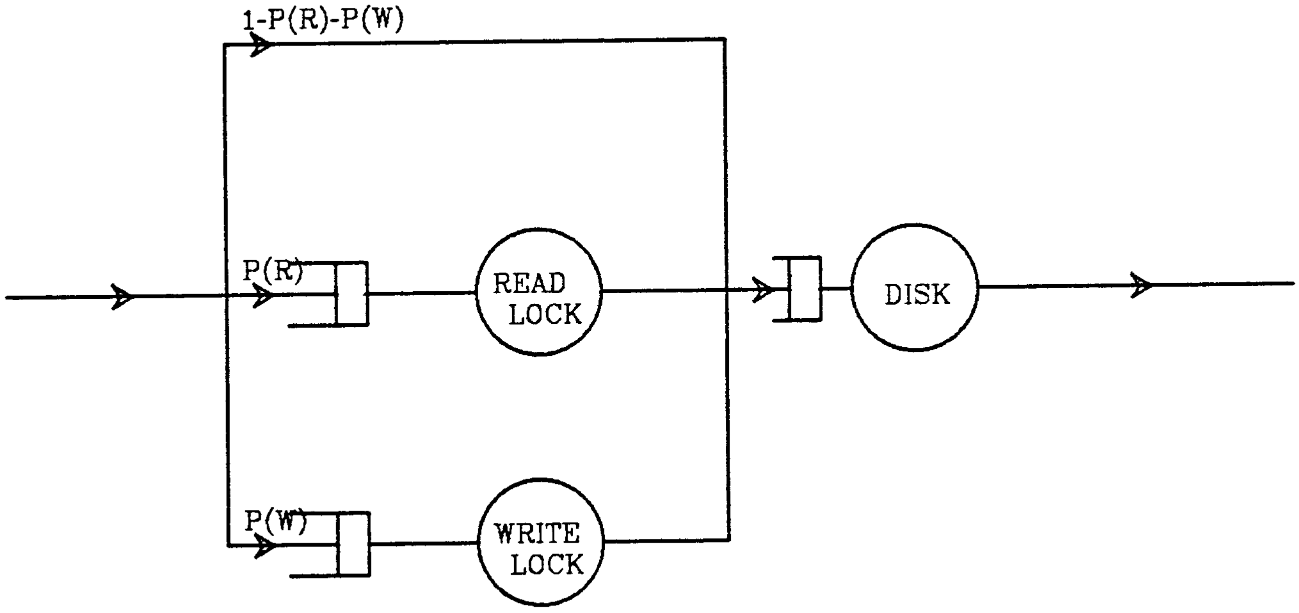


Fig. 3. The analytic model of a file disk.

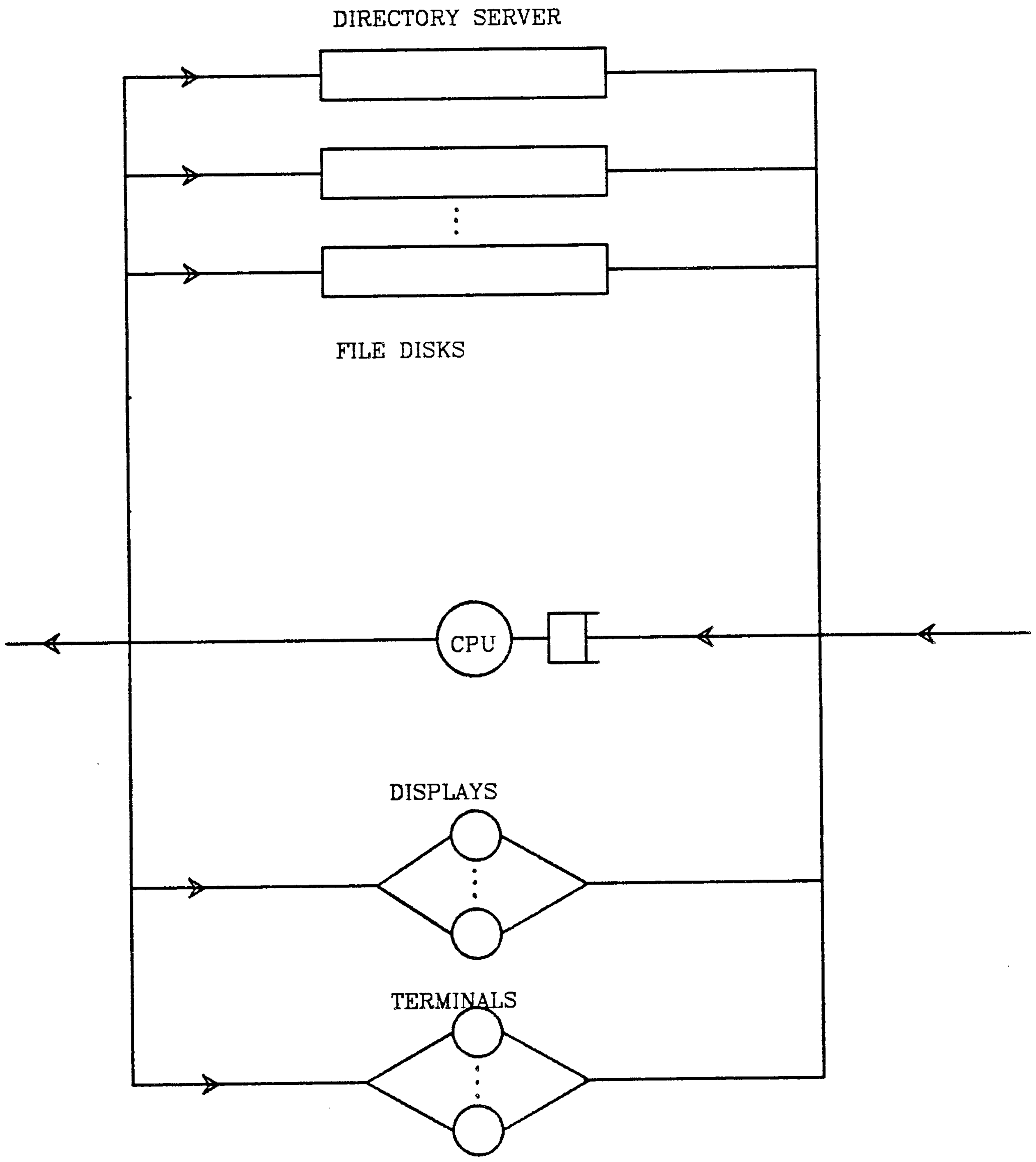


Fig. 4. The model of a host.

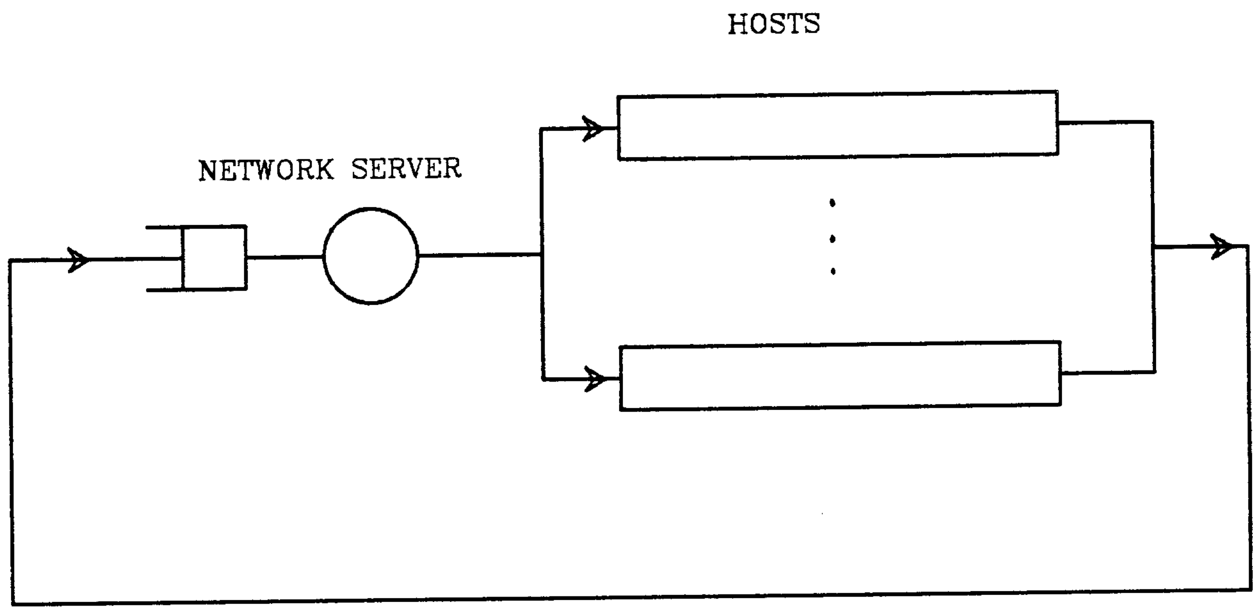


Fig. 5. The model of a distributed system.