Modeling Process-Resource Activity*

by

Janis P. Osterweil
and
Gary J. Nutt
Department of Computer Science
University of Colorado
Boulder, Colorado 80309

TR #CU-CS-084-75

November, 1975
Revised March, 1978

* This work was partially supported by the National Science Foundation under Grant No. MCS74-08328.

ABSTRACT

Modeling Process-Resource Activity

Janis P. Osterweil
and
Gary J. Nutt

Many models are present in the literature that have been used for investigating deadlock avoidance, prevention, and detection algorithms. In this paper, a model is presented that concentrates on possible conditions that may exist in a system of n processes and k resource types with varying units of resources. The model is based on a graph in which each node represents a state of the entire system. It can be shown that this set of nodes can be partitioned into three sets such that all nodes in the first set represent the condition that the corresponding system is guaranteed to be free of deadlock; the second set represents the condition that the system is deadlocked; the third set represents the condition that the system may or may not be in a state of deadlock, depending on how each given node has been reached from the initial node of the graph. The model has been exercised with various system configurations to obtain probabilistic estimates of deadlock existing in each node of the third set, and these results are given, as well as a description of the model.

Key words: deadlock, simulation, detection, deadlock probability

CR categories: 4.35,8.1

INTRODUCTION

Deadlock has been the subject of a significant amount of research in the last several years, beginning with the work of Dijkstra [3] and continuing to the present day. Many solutions to the deadlock problem exist for a variety of systems, [1,5,6], but few of these solutions are actually implemented in real operating systems. There are two reasons for this: first, the implementation is costly both in the space required to maintain tables indicating the current status of the system, and the amount of processing time required to keep the state of the system current and to detect or avoid a deadlock state [2]; second, the frequency of occurrence of deadlocks in current systems has not made the implementation cost-effective from a management point of view [1]. The deadlock problem can be expected to become more serious as the number of processes and the number of resources in a system increase [1,4]. Thus, if computer systems continue to grow to systems with many processes and resources, one should carry out more intensive investigations with respect to the expectation of deadlock in these systems. To date, little data concerning the characteristics of deadlock has been gathered either from actual operating systems or from simulation studies.

In this paper, a pragmatic graph model for studying a family of systems with varying numbers of processes, types of resources, and numbers of each type of resource is presented. Other graph models have been used to investigate deadlock in operating systems, (e.g., [6]), although most of these models represent the state of the system with respect to the activities of processes. The model discussed in this paper represents the state of resources--the number of available resource units, the number of unfulfilled requests. The graph representation is static and does not

change its topology as process activity takes place. The nodes of the graph are isomorphic to points in a three dimensional Euclidean space, where each dimension is bounded by the number of resource types, the number of units of each type, and the number of processes in the system.

Although the model is useful for obtaining analytic results, here it is used only to explain a simulation study of processes and resources. The model characterizes resource activity in the class of single unit request operating systems, allowing one to investigate the probability that deadlock will exist under varying conditions. If a process requires multiple units of a resource, then successive requests can be issued, (each request after the first being preceeded by an allocation). This information is useful in algorithmically determining the particular set of circumstances under which deadlock detection should be invoked, or even if a detection or avoidance algorithm should be employed in an environment determined by the number of processes, units of resources, and resource types.

In the remainder of the paper, the class of operating systems that can be represented is first described. Next, the basic graph model is described in detail, including its interpretation and some examples. The experiment is then discussed, followed by a summary of the results. Finally, the conclusion provides criticism of the model and the method, and then indicates the utility of the results given in this paper.

A more complete treatment of the topics discussed here appears in [7].

THE CLASS OF SYSTEMS

The model represents systems that are composed of k distinct types of resources, with m_g units of the type g resource, $1 \leq g \leq k$. The model provides for n processes all competing for resource units. For example, if a system is configured with 3 line printers (resource type 1), 2 card readers (resource type 2), and 4 allocatable partitions of memory (resource type 3,)then $k=3, m_1=3, m_2=2, m_3=4$.

It is assumed that a process will never request more than one unit of any resource at a time, and that another request cannot be made until the first has been satisfied by allocation, i.e., the system supports only single unit requests; it is also required that a process will deallocate its resources upon completion. A process will not request more units of any resource than exist in the system. Non-preemption of resources, the possibility of a circular wait, and exclusive control of a unit of resource are assumed, since the model must allow deadlock conditions to exist in order to investigate the circumstances under which deadlock is likely to occur.

A GRAPH MODEL FOR RESOURCE ACTIVITY

The directed graph model, $G_{nm_1m_2\ldots m_k k}$, is used to model a system composed of n processes, k different resource types, such that m_i represents the number of units of resource type i. The model consists of the vertex set V and the edge set $E = E_r \cup E_a \cup E_d$, where

$$V = \{(i,j,g) \mid 0 \le i \le n,\ 0 \le j \le m_g - 1,\ 1 \le g \le k\} \cup \{(0,m_g,g) \mid 1 \le g \le k\}$$

$$E_r = \{((u_1,u_2,g),(v_1,v_2,g)) \mid v_1 = u_1 + 1,\ v_2 + u_2 \text{ for } 0 \le u_1 \le n-1,\ 0 \le u_2 \le m_g - 1,\ 1 \le g \le k\}$$

$$E_a = \{((u_1,u_2,g),(v_1,v_2,g)) \mid v_1 = u_1 - 1,\ v_2 = u_2 + 1 \text{ for } 1 \le u_1 \le n,\ 0 \le u_2 \le m_g - 2 \text{ or else } u_1 = 1 \text{ and } u_2 = m_g - 1, \text{ and } 1 \le g \le k\}$$

$$E_d = \{((u_1,u_2,g),(v_1,v_2,g)) \mid v_1 = u_1,\ v_2 = u_2 - 1 \text{ for } 0 \le u_1 < n \text{ and } 1 \le u_2 \le m_g - 1 \text{ or else } u_1 = 0 \text{ and } u_2 = m_g, \text{ and } 1 \le g \le k\}.$$

Less formally, the set of vertices roughly correspond to the allocation status of a system at any given time. The ordered triples, (i,j,g) define a discrete three dimensional space where the particular point (i',j',g') represents the fact that there are i' outstanding requests for units of resource type g',

while j' of those units have previously been allocated to some processes. Thus the plane, g, is a state transition diagram for resource type g. The edge sets (E_r,E_a, and E_d) represent allowable state transitions within each of the g transition diagrams. Some examples can clarify the discussion.

Figure 1 is a representation of the graph $G_{nm_1m_2m_3k} = G_{33243}$--a system of n = 3 processes, k = 3 resource types, m_1 = 3 units of resource type 1, m_2 = 2 units of resource type 2, and m_3 = 4 units of resource type 3 (similar to the system mentioned in the previous section). The graph model can be used to describe resource allocation and deallocation under the following interpretation: All edges which belong to the set E_r are labelled "r_g" for $1 \leq g \leq k$ which denotes a request of resource type g; all edges which belong to E_a are labelled "a_g" for $1 \leq g \leq k$ which denotes an allocation of a unit of resource g; and all edges which belong to E_d are labelled "d_g" for $1 \leq g \leq k$ which denotes a deallocation of a unit of resource type g. We are only interested in the paths initiating from vertex (0,0,g), for all g where $1 \leq g \leq k$ (corresponding to the state transition for each resource type). The set of all paths from vertex (0,0,g) back to vertex (0,0,g) represents the set of possible actions, taken by the resource scheduler and the n processes, which are guaranteed to avoid deadlock regardless of the processes involved in each action, [7].* In the example shown in Figure 1 (G_{33243}), a path is represented by listing the edge labels on the path. For example, "$r_2r_2a_2d_2a_2a_2d_2$" represents a traversal from node (0,0,2) through (1,0,2), (2,0,2), (1,1,2), (1,0,2), (0,1,2), to (0,0,2). Concurrent path traversals are represented by interleaving symbols with different subscripts, e.g. "$r_1a_1r_2r_3a_3d_1a_2d_2d_3$" represents request, allocation, and deallocation of each resource type by anonymous processes.

---

* Node (n,m_g-1,g) represents deadlock freedom although there is no path from this node back to (0,0,g). The node allows cycles in the augmented graph discussed in the next paragraph, where the semantic interpretation given here is consistent.

$G_{nm_1m_2\ldots m_k k}$ consists of $\sum_{g=1}^{k} ((n+1)*m_g + 1)$ vertices, where each vertex is labelled to indicate the number of single unit requests and allocated resource units of resource type g represented by the vertex. Thus vertex (p,q,g) represents p requests for a resource unit of type g which have not yet been allocated, and q resource units of resource g which are currently allocated to a subset of the n processes. Notice that the single unit request assumption allows one to bound the value of i in (i,j,g), since it permits a maximum of one outstanding request per process. The extension to multiple unit request models increases the size of the graphs substantially.

Although the set of cycles through (0,0,g) in $G_{nm_1m_2\ldots m_k k}$ represents sequences of actions that are ensured of being free from deadlock, it is also desirable to be able to represent sequences of actions in which deadlock is possible. $G_{nm_1m_2\ldots m_k k}$ can be modified to include these other paths by adding vertices corresponding to (i,m_g,g) for $1 \le i \le n$ and for all g, $1 \le g \le k$. Figure 2 is the extended graph, $G'_{nm_1m_2\ldots m_k k}$, corresponding to Figure 1. These additional vertices allow partial paths to exist that may correspond to deadlock, depending on which particular processes performed the requests and acquired units of the resource. In the unaugmented model, process identity is irrelevant, but in the model which includes vertices representing deadlock states, the identity of process which cause resource state changes is pertinent. The vertex labelled (n,m_g,g) for any g represents a situation in which all n processes have a pending request for a unit of resource type g and all m_g units are currently allocated. Hence, any partial path leading from (0,0,g) to (n,m_g,g) represents a sequence of resource requests and allocations that is guaranteed to put the system in a state of deadlock.

Consequently, there is no exit edge from vertex (n,m_g,g) since resource preemption is not allowed, and there exists no path from $(0,0,g)$ to $(0,0,g)$ that passes through vertex (n,m_g,g).

Now consider the set of vertices

$$\{(i,m_g,g) \mid 0<i<n, \quad 1\leq g\leq k\}.$$

Each of these vertices represents a situation in which all m_g units of the resource g are allocated and there exists i outstanding requests for the resource (by i distinct processes, since only single unit request systems are represented by the model). Thus the possibility of deadlock exists in those states, depending on the identity of the processes which changed the resource state into (i,m_g,g). Let superscripts identify the process associated with a resource activity (subscripts still identify the resource type), and consider the graph G'_{33243} of Figure 2. The string "$r_1^1 r_1^2 a_1^1 r_1^3 a_1^2 a_1^3 r_1^1 r_1^2$" corresponds to a path from (0,0,1) to (1,0,1,) since process 1 requested a unit of resource type 1 (r_1^1), then to (2,0,1) since process 2 requested a unit of resource type 1 (r_1^2), then to (1,1,1) since process 1 was allocated a unit of resource type 1 (a_1^1), etc. to (2,3,1). In this situation, process 3 has been allocated one unit of resource 1 (a_1^3) but it is not blocked on resource 1 nor resource 2, i.e., the system is not in a state of deadlock. Alternatively, the string "$r_1^1 r_1^2 a_1^1 r_1^1 a_1^2 a_1^1 r_1^1 r_1^2$" in the same configuration traverses the same path from (0,0,1) to(2,3,1), but in this case processes 1 and 2 are deadlocked since all three units of the resource are allocated to processes 1 and 2 and these two processes have outstanding requests which can never be satisfied (unless preemption is allowed). From these two examples, it can be observed that the vertices (i,m_g,g), $1\leq g\leq k$, for $0<i<n$, correspond to situations in which deadlock may exist,(depending on the processes involved) but it is

not guaranteed to exist as it is at vertex (n,m_g,g).

It is also possible for a path such as "$r_1r_1a_1r_1r_1a_1a_1r_1$" to lead to vertex (2,3,1) in the example; however, the sequence of activities will never represent a state of deadlock on resource 1 under any process numbering. The key points in recognizing that this statement is true are that only single unit requests are allowed in the system and that a process cannot request more than m_g units of resource type g.

The model is used to identify the possibility of deadlock across resource types, as well as within a given resource type as discussed above. For example, the sequence "$r_1^1r_1^2a_1^1r_2^1r_1^3a_1^2r_2^2a_2^1a_1^3r_2^3a_2^2r_2^2r_1^1$" describes a path leading from (0,0,1) to (1,3,1) = $(1,m_1,1)$ and another path leading from (0,0,2) to (2,2,2) = $(2,m_2,2)$ in the system shown in Figure 2. Now, the sequence indicates that process 1 is requesting a unit of resource type 1 while holding a unit of type 1 and a unit of type 2; process 2 is requesting a unit of type 2 while holding a unit of type 1 and a unit of type 2; process 3 is requesting a unit of type 2 while holding a unit of type 1. The system is deadlocked across resource types, even though the sequence of actions only on type 1 or only on type 2 would indicate blocking and not deadlock. It has been shown that this deadlock across resource types can only exist when paths in two or more of the g (sub)graphs simultaneously lead to $(i,m_{g'},g')$ and $(j,m_{g''},g'')$ for $i \neq 0$ and $j \neq 0$, [7].

The critical issue in using the graph model is the determination of deadlock existence when a path leads to one of the vertices (i,m_g,g) in one of the parallel (sub)graphs.* Since these vertices sometimes represent

---

* In the special case of a single resource system, (k = 1), vertex (1,m,1) is always a non-deadlock vertex since deadlock cannot exist on a single resource type with only one pending request for an unavailable unit of the resource.

a state of deadlock, it is desirable to determine either the circumstances under which a path causes deadlock in one of these gray vertices, or to be able to attach a probability that deadlock exists whenever a path leads to one of the vertices. No set of sufficient conditions have been found to shed light on the problem; instead, the concentration has been on determining probabilities of deadlock for the gray vertices. The activity of a set of systems was simulated under varying conditions and probabilities of deadlock for each gray vertex were obtained.

THE EXPERIMENT

The primary reason for simulating the process/resource activity is to determine the probability that the system is in deadlock when at least one path in one of the k parallel graphs leads to a vertex (i,m_g,g) for $0<i<n$, $1\leq g\leq k$. The set of parameters that vary for different tests is the number of processes, n; the number of types of resources, k; and the number of units of each type of resource, m_g (for $i\leq g\leq k$). Additionally, the simulation program can indicate other properties of the configuration:

- Under varying conditions, how does the mean number of action steps, (i.e., requests, allocations, and deallocations), to deadlock change?
- How does the frequency with which processes request and deallocate resources affect the probability of that process being involved in a deadlock?

The approach used in defining the test cases was to specify n, m_g, and k; for each process, define a frequency rate, (based on a Poisson distribution), for resource request and deallocation; specify a maximum number of trials in the test, where each trial ends by the system coming

to a deadlock state or some maximum number of requests and deallocations have taken place. During each test run, the number of times that each gray vertex was entered was recorded, the number of times that deadlock was detected was recorded, and in the case of deadlock existence, the number of activity steps leading to the deadlock was recorded. From this information, the probability of deadlock of each vertex, (i,m_g,g) was computed and the mean number of activity steps to deadlock was available.

Thirty cases were tested where each case was composed of 1,000 trials. Most trials allowed a maximum of 100 to 200 steps, causing deadlock to be encountered about 25,000 times. The summary of all of these tests is given in reference [7] and in the next section a few cases are described.

RESULTS

First, Figure 3 shows how the probability of deadlock varies as n changes; this set of test cases uses 3 resource types, with 3 units of each resource. Since there are 3 vertices corresponding to (i,m_g,g) for each i, the average probability of deadlock in all 3 graphs is plotted. The unorthodox abscissa labelling is required to allow easier comparison of curves under varying values of n, i.e., the probability that deadlock exists when at (i,m_g,g) is one if $i=n$ for all values of n. In order to compare the probabilities for varying n, the abscissa origin is adjusted as a function of n. Two conclusions can be drawn from this figure (and others not given in this paper due to space limitations): The probability of deadlock increases with i for the vertex labelling, (i,m_g,g). The probability of vertex $(n-i,m_g,g)$ being deadlock instance increases as the number of processes, n, increases.

In Figure 4, data are plotted to show the probability variation as the number of resources of a single type is increased. Again, the probability of deadlock is greater for vertex $(i,m_1,1)$ than for $(i',m_1,1)$ whenever $i>i'$.

Although the probability of vertex $(i,m_1,1)$ being in deadlock increases as the number of units of the single resource type decreases in most cases, two contradictions occur at vertex $(n-1,m_1,1) = (8,m_1,1)$. No explanation for these discrepancies has been discovered. In each case, however, there is a sharp increase in deadlock probability at node $(n-1,m_1,1)$ relative to the previous node $(n-2,m_1,1)$. Also, rarely does node $(n-1,m_1,1)$, for any value of n or m_1, exceed 0.5 probability and yet node $(n_1m_1,1)$ represents a deadlock probability of 1.0.

Data was gathered from 8 out of the 30 cases with more than one resource type ($k>1$) to indicate the percentage of deadlock occurrences involving only one resource type. In most cases (6 out of 8) more than 50% of the deadlocks that occurred involved one resource type only. (The average percentage of deadlocks involving one resource type over all 8 cases was 57.3%.) Thus it appears to be worthwhile to study the deadlock situation on a resource system of only one resource type.

Figure 5 plots the number of processes, n, versus the mean transition to deadlock; it illustrates two things. First, as resource units (m_g) are added to a resource type (g), the probability of deadlock decreases, irrespective of the number of processes (n) involved. Second, the figure illustrates the mean number of activity steps that were taken before a deadlock was encountered. under varying number of processes. The data indicates that this mean number increases with the number of processes, n. The reason for this apparent anomaly, (i.e., one might expect the number of activity steps to decrease with increasing n), is that deadlock can can occur only when processes request resources at a time that they already hold other resources. As the number of processes increases, the likelihood of this circumstance occurring decreases, and the mean number of activity steps increases. Several other test cases not shown in the figure also yielded the same result, even when the number of resource

types was increased k>1), and the numbers of units of each type differed.

Figure 6 shows the effect of uniformly adding processes (n) and units of one resource (m_g) to a system; the mean number of transition steps to deadlock increases with n and m_g.* Hence in a fixed resource system of one resource type, if the number of processes acting on this system increases and/or more units of the resource are added, then the probability of deadlock decreases.

The frequency of process requests and deallocations was used as an input parameter to the model. For each process, i, ($1 \le i \le n$) R_i specifies a request frequency and D_i specifies a deallocation frequency, where $0<R_i<1$ and $0<D_i<1$. The time interval between a request or deallocation is Poisson distributed; the values of R_i and D_i determine the choice of request or deallocation as follows:

1. Generate a random number, x, from a uniform distribution between 0 and 1.
2. If

$$x < \frac{\sum_{i=1}^{n} R_i}{\sum_{i=1}^{n} (R_i+D_i)}$$

then the event is a request. Otherwise it is a deallocation.

Now, large R_i relative to D_i cause more frequent allocation than deallocation.

Figure 7 plots R_i against the probability that any given process will be involved in deadlock for a system of 9 processes with 5 resource types, each with 9 units of resource. Processes which request resources frequently are often involved in deadlock, while processes that require

* Total transitions include requests, deallocations, and allocations; process transitions are only requests and deallocations.

resources less frequently are involved in deadlock much less often. It is of interest to note that the processes which deallocate resources frequently are involved in deadlock only about 10% less often than those processes which deallocate resources less frequently, irrespective of the request frequency of the processes (this can be seen by a vertical inspection of the graph of figure 7). Hence it appears that requests correlate to a deadlock condition more than deallocation.

CONCLUSIONS

The graph model described in this paper has been used to describe the status of an operating system with respect to deadlock. The vertices of the model are partitioned into classes such that the first class represents system states that are free from deadlock regardless of the path leading to that vertex; the second class represents system states in which it is certain that deadlock exists; the third class represents system states where deadlock might exist.

Although one might be able to obtain a closed form solution to determine the probability of deadlock for each vertex in this latter class, estimates of those probabilities have been obtained for a variety of system parameters by simulation.

The primary criticism of the model might be that it only applies to single unit request allocation strategies. The model could be extended to handle multiple unit requests, but many of the results would no longer hold and the model may become intractable. A simple extension to the model would allow single unit requests for each resource type to be pending.

Other criticisms have to do with the simulation technique: no scheduling algorithm was explicitly modeled; instead, Poisson distributions were used to generate random scheduler activity. This approach was taken in order to avoid

special-case studies involved with each scheduling algorithm. Thus the results may be more pessimistic then might be experienced with particular schedules. Similarly, there was no system monitoring to generate resource request (deallocation) sequences in deriving the load on the resource system; Poisson distributions were used to simulate random demand in order to obtain general measures that did not reflect particular workloads.

The significant results from the study indicate that a deadlock detection algorithm should be invoked with every resource request when the system reaches a point of having n-h, $0 \leq h \leq n$, outstanding resource requests for a given resource type, and that as h decreases, the invokation of the detection algorithm becomes more and more important. There is little need to invoke the detection algorithm if all units of the given resource type are allocated and there is only a relatively small number of outstanding requests.

A surprising result is that the number of resource types involved in deadlock, in a multiple resource system, is frequently only one. Thus, in a computer system where deadlock may be particularly disastrous for some single resource type, (e.g., tape drives), it is worthwhile to detect or avoid deadlock on that particular resource even if others are ignored.

REFERENCES

[1] Coffman, E. G., Elphick, M. T., and Shoshani, A., "System Deadlocks", ACM Computing Surveys, Vol. 3, No. 2, (June, 1971), pp. 67-78.

[2] Denning, P. J., "Third Generation Computer Systems", ACM Computing Surveys, Vol. 3, No. 4, (Dec., 1971), pp. 175-216.

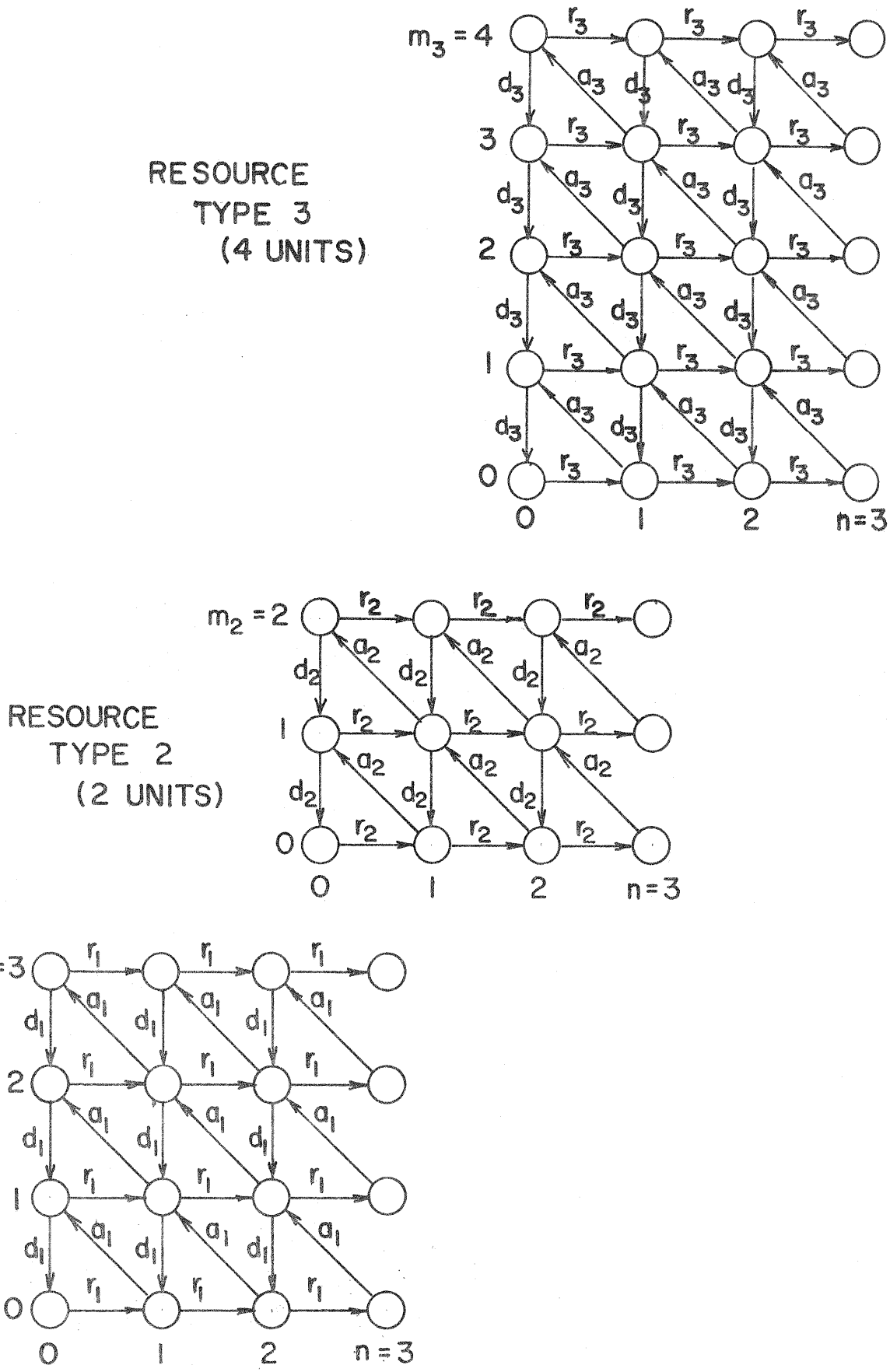
[3] Dijkstra, E. W., "Co-operating Sequential Processes", In Programming Languages, F. Genuys (Editor), Academic Press, (1968), pp. 43-112.

[4] Ellis, C. A., "On the Probability of Deadlock in Computer Systems", Proceedings of the Fourth Symposium on Operating Systems Principles, Yorktown Heights, NY, (1973), pp. 88-95.

[5] Havender, J. W., "Avoiding Deadlock in Multi-tasking Systems", IBM Systems Journal, Vol. 2 (1968), pp. 74-84.

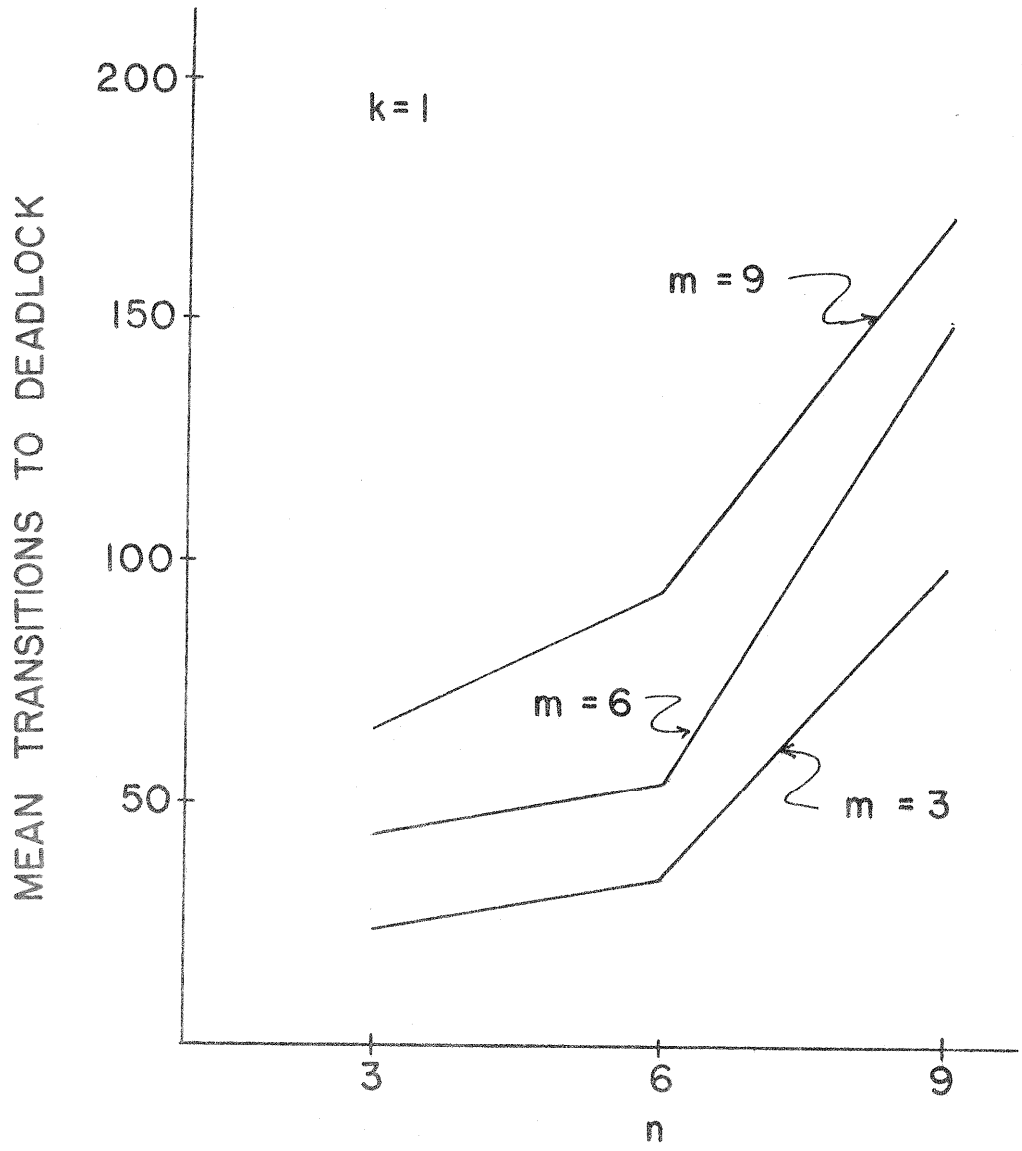
[6] Holt, R. C., "Some Deadlock Properties of Computer Systems", ACM Computing Surveys, Vol. 4, No. 3, (Sept., 1972), pp. 179-196.

[7] Osterweil, J. P., "A Deadlock Model Based on Process-Resource Actions", M. S. thesis, Department of Computer Science, University of Colorado, (1975).



$$G'_{n m_1 m_2 m_3 3} = G'_{33243}$$

FIGURE 2

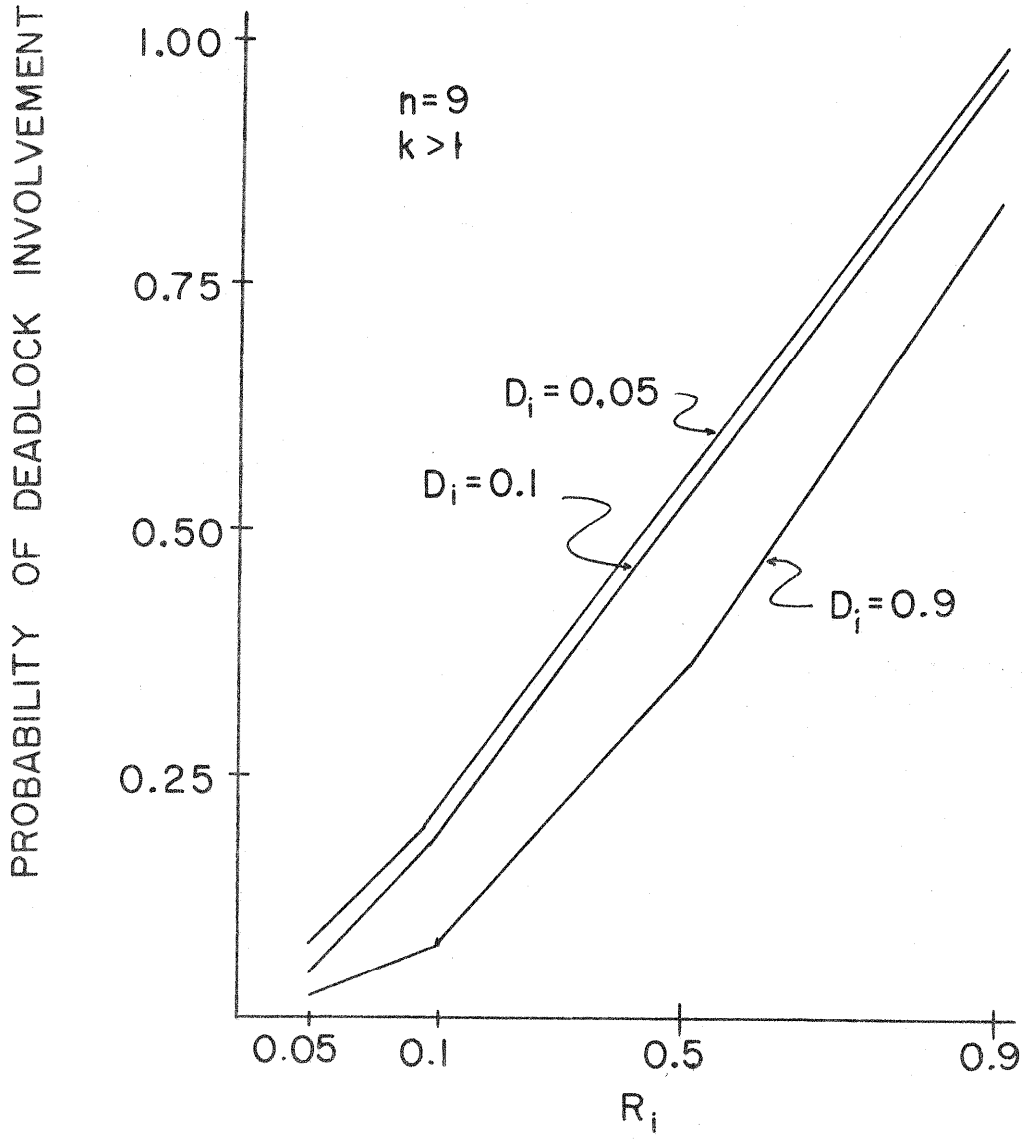


Caption:

k = number of resource types
m = number of units of the single resource
n = number of processes

MEAN STEPS TO DEADLOCK UNDER VARYING m OR n

FIGURE 5



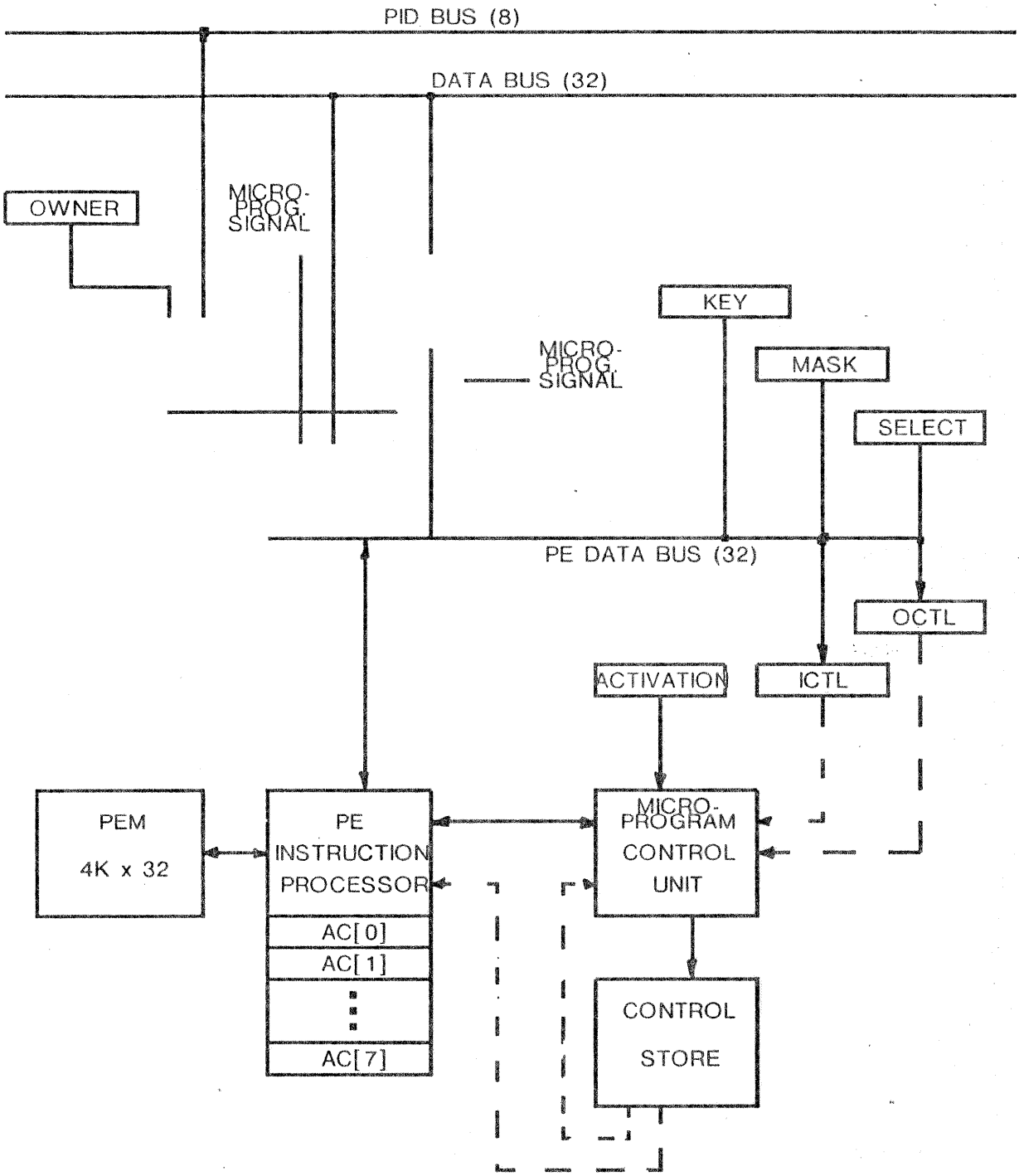
DEADLOCK PROBABILITY WITH RESPECT TO R_i & D_i

FIGURE 7

Caption:

R_i - request frequency for process i

D_i - deallocation frequency for process i.



PROCESSING ELEMENT ORGANIZATIONff

FIGURE 3