Vijay Ahuja

Determining Deadlock Exposure for a Class of Store and Forward Communication Networks

Given a network design, is the network exposed to deadlock for message buffers? This problem is addressed for a class of networks called "routed networks." A routed network is a store and forward communication network in which all transmissions follow predefined routes through the network. It is shown that a routed network is exposed to deadlock if and only if there exists a complete weighted matching of an appropriately defined bipartite graph for some subgraph of the network graph. An approach for determining whether a deadlock can occur is presented.

1. Introduction

Consider a store and forward communication network in which the task of transmitting messages between the nodes is accomplished by moving the message through various transmission links and message buffers along fixed routes. A message is stored and then transmitted to the next node on the route. Transmission of a message from a node does not start until a buffer at the next node on the route has been allocated to it. A deadlock will occur in such a network if two or more messages, collectively occupying all the buffers in some nodes, wait for each other to release a buffer. Figure 1 illustrates a store and forward communication network with one buffer in each node. Figure 2 illustrates a deadlocked condition in this network, since each task has one buffer but requires two, one at each end of the transmission link.

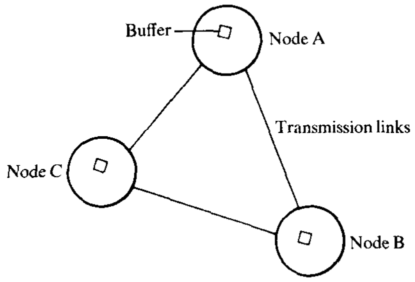


Figure 1 A store and forward communication network.

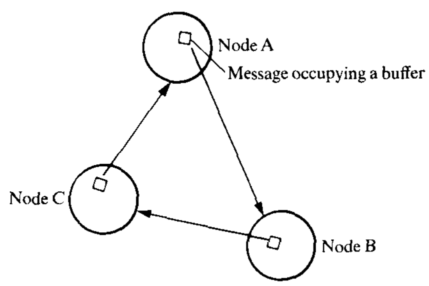
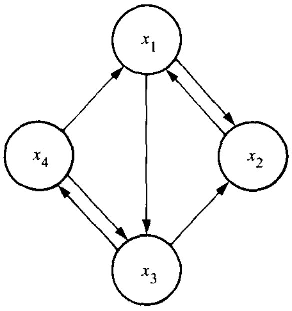


Figure 2 A deadlocked condition of the network in Fig. 1. (The direction of the arrow indicates that the message on its trailing end is waiting for a buffer in the node at its leading end.)

The general problem of predicting deadlock in a system has been treated by several authors [1-3]. The problem can be simplified if it is restricted to a single class of systems and a single class of resources. In this paper, we restrict the problem to the class of store and forward communication networks that have fixed routes for message transmission between any two nodes [4]. The only class of resources treated here is the message buffers at the nodes of the networks.

The above problem is similar to detecting the existence of direct and indirect store and forward deadlocks in the



$N = (X, A, B, R, M)$

X, A: The set of nodes and directed arcs as shown above.

B: $\{b_i = 2 | i = 1 \text{ to } 4\}$

R: $\{r_1, r_2, r_3\}$

$r_1 = (x_4x_1, x_1x_2)$

$r_2 = (x_1x_3)$

$r_3 = (x_3x_4)$

M: $\{m_i = 2 | i = 1 \text{ to } 3\}$

Figure 3 A routed network.

ARPA network described in [5, 6]. The difference, however, lies in our restriction of fixed routing as compared to the adaptive routing in the ARPA network. Our model for network deadlocks does not allow any actions taken by the networks to prevent the congestion that usually precedes a deadlock. Raubold *et al.* [7] have developed a buffer allocation scheme that prevents occurrences of store and forward deadlocks. Our approach does not impose any buffer allocation restrictions on the network operation. Instead, it examines the network design to determine whether a deadlock can occur. Finally, as we point out later, any deadlock state detected by these results may not be reachable.

In Section 2 of this paper, a graph model is developed for the class of store and forward communication networks described above. In Section 3, the problem of determining whether such a network is exposed to deadlock is transformed to a matching problem. In Section 4, results from network flow theory are used, and an approach to solve the problem is provided.

2. A graph model

The nodes, links, and buffers in a communication network can be represented by a directed graph in which each node is labeled with the number of buffers in it. The graph should have no self-loops, since we do not wish to consider links that initiate and terminate on the same node. So a *network graph* $G = (X, A, B)$ is a triple, where X is a finite nonempty set representing the nodes of the network, A is an irreflexive relation $X \times X$, representing the directed links of the network, and B is a function on X to the set of positive integers: $B(x_i)$, or simply b_i, represents the number of buffers in node x_i.

A *route* of a network graph $G = (X, A, B)$ from node x_a to node x_b is simply a directed path from x_a to x_b in G.

Our definition for a network also includes a maximum number of messages allowed on each route at any time. In this context, a *message* is simply a unit of data that occupies one buffer. Thus, a *routed network* is a quintuple $N = (X, A, B, R, M)$, where (X, A, B) is a network graph G, R is a set of routes of G, and M is a function on R to the set of positive integers; $M(r_i)$, or simply m_i, represents the maximum number of messages allowed on route r_i at any time.

Routed networks have fixed routes and allow no more than a fixed number of messages on each route at any time. (There may be more than one fixed route between a given pair of nodes.) A direct analogy is a fixed-route network that employs end-to-end flow control permitting a fixed number (m_i) of messages on each route (r_i).

In order to define a deadlock condition in a routed network, we need to describe its activity in terms of "tasks" and "states." A task, as defined next, represents a message and can be identified by the route and an index. The index distinguishes a given task from other tasks on the same route. So, in a routed network $N = (X, A, B, R, M)$, a *task* is defined by a pair $\langle r_i, p\rangle$, where r_i is a route in N and p is a positive integer less than or equal to the number of messages m_i allowed on r_i.

Now, once a message reaches its destination node, it does not require any more buffers for transmission. A state maps a given task to a node, other than the last node, on its route. Furthermore, the number of tasks mapped to a given node cannot exceed its number of buffers. So, a *state* S of a routed network $N = (X, A, B, R, M)$ is a function

S: $T \rightarrow (X \cup \{z\})$, such that

1. For each task $t = \langle r, p\rangle$ in T, if $S(t)$ is in X, then $S(t)$ is a node, other than the last node, in r.
2. For each node x_i in X, the number of tasks in $\{t | S(t) = x_i\}$ is not greater than the number of buffers b_i in x_i.

Here, T is the set of all tasks of N,

$T = (\langle r_1, 1\rangle, \cdots, \langle r_1, m_1\rangle, \cdots, \langle r_{|R|}, 1\rangle, \cdots, \langle r_{|R|}, m_{|R|}\rangle)$,

and z is a fictitious node, not in X.

The state function maps those tasks to z which do not exist in the network at that time.

We can now describe the deadlock condition of a network. We begin by observing that in order to transmit a message to its destination node, a task requires a buffer in the next node along its route. So, consider a state, say D, in which the tasks in a set $T' \subseteq T$ are waiting for each other for a buffer. Furthermore, there are no free buffers in any node with one or more tasks of T'. Clearly, such a wait is not resolvable and therefore implies a deadlock. So, a *deadlock* or a *deadlock state* is a state D of a routed network $N = (X, A, B, R, M)$, in which there exists a nonempty set $T' \subseteq T$ of tasks such that for each task $t = \langle r, p \rangle$ in T',

1. The next node, say x_k, after $D(t)$ in route r, is in the set $\{D(t_i) | t_i \in T'\}$;
2. The number of tasks in $\{t_i \in T'$ and $D(t_i) = x_k\}$ is equal to the number of buffers b_k in node x_k.

The set $\{x | x = D(t)\}$ of nodes is called a *deadlock set*. Finally, a routed network is *exposed to deadlock* if it has a deadlock state. (We do not address the problem of determining whether a given deadlock state is "reachable," *i.e.*, it can be reached by some task sequence from an initial state.)

An example of a routed network is illustrated in Fig. 3. Consider the state D of this routed network that maps the two tasks of route r_1 to node x_4, those of r_2 to x_1, and those of r_3 to x_3 (Fig. 4). Then each task residing in a node in the set (x_1, x_3, x_4) of nodes requires a buffer from another node in this set, and there is no free buffer. So D is a deadlock state, and the set (x_1, x_3, x_4) of nodes is a deadlock set.

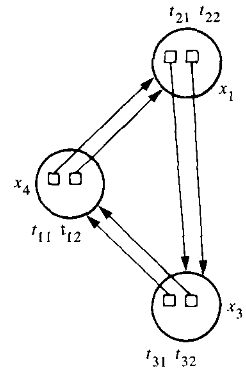


Figure 4 A deadlock state in the routed network of Fig. 3. (An arrow indicates that the task on its trailing end requires a buffer in the node at its leading end.)

3. Assignment of tasks to buffers

In order for a given set X' of nodes to be a deadlock set in a deadlock state D, there must exist a set T' of tasks that occupy each buffer in X'. Also, for each task $t = \langle r, p \rangle$ in T', the next node in route r after node $D(t)$ must be in X'. This can be established by matching eligible tasks to buffers in the given set of nodes. This approach of matching tasks to buffers through bipartite graphs permits the use of graph theory results in examining networks for deadlock exposure. Before describing this approach, we present some pertinent graph theory definitions.

In a graph $G = (X, E)$, two nodes x, y in X are *adjacent* if there exists an edge $(xy) \in E$. A *bipartite graph* $H = (X_1, X_2, E)$ is an undirected graph $(X_1 \cup X_2, E)$ whose nodes have been partitioned into two disjoint sets, X_1 and X_2, whereby no two nodes in any one of the subsets are adjacent.

Consider the subgraph $S_g = (X', A')$ generated by $X' \subseteq X$ on a network graph $G = (X, A, B)$. It is to be determined if X' is a deadlock set. We represent the subgraph S_g and the tasks associated with it by a bipartite graph. One set of nodes of the bipartite graph represents the nodes of S_g and the other set of nodes represents the set of routes. The edges of the bipartite graph represent each task that can have a buffer allocated in some node of S_g and requires a buffer in some other node in it. Thus, node x_i is connected to route r_j only if x_i is a node on r_j and the next node on r_j, after x_i, is in S_g. This is defined next.

Let $S_g = (X', A')$ be the subgraph generated by $X' \subseteq X$ on a network graph $G = (X, A, B)$ of a routed network $N = (X, A, B, R, M)$. Then we say that $H = (X', R, E)$ is the *bipartite graph for the subgraph* S_g, where E is the set of edges such that the number of edges connecting a node $x_i \in X'$ and a route $r_j \in R$ is

1. Min (b_i, m_j) if node x_i is on route r_j and the next node after x_i, on r_j, is in X';
2. Zero otherwise.

Next, a "weighted matching" of such a bipartite graph is defined. Any matching of tasks to buffers must correspond to a feasible state. Since the edges of a bipartite graph represent the tasks, there cannot be more than b_i edges connected to a node x_i and no more than m_j edges connected to a route r_j. Let $H = (X', R, E)$ be the bipartite graph of a subgraph $S_g = (X', A')$ generated by $X' \subseteq X$ on a network graph $G = (X, A, B)$ of a routed network $N = (X, A, B, R, M)$. Then a *weighted matching* of the *bipartite graph* H is a set $E_0 \subseteq E$ of edges such that

1. For each node $x_i \in X'$, there are no more than b_i edges in E_0 connected to x_i, and

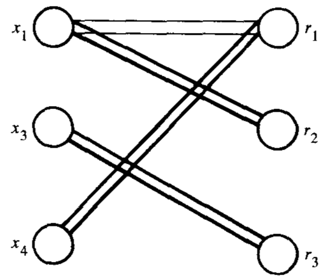


Figure 5 A maximum weighted matching of the bipartite graph. $H = (\{x_1, x_3, x_4\}, R, E)$ generated by $X' = (x_1, x_3, x_4)$ on the network graph of Fig. 3.

2. For each route $r_j \in R$, there are no more than m_j edges in E_0 connected to r_j.

A weighted matching E_0 of a bipartite graph $H = (X', R, E)$ is *complete* if, for each node $x_i \in X'$, the number of edges in E_0 connected to x_i is b_i. Finally, a *maximum* weighted matching E_0 of a bipartite graph $H = (X', R, E)$ is a weighted matching such that if E_i is a weighted matching of H, then $|E_i| \leq |E_0|$. A maximum weighted matching is not necessarily unique. Also, it can be readily seen that a complete weighted matching is also maximal. Furthermore, a complete weighted matching is not necessarily unique, but if one maximum weighted matching is complete, then every maximum weighted matching is complete. An example of a weighted matching for a bipartite graph $H = (X', R, E)$ generated by $X' = (x_1, x_3, x_4)$ on the network graph of Fig. 3 is illustrated in Fig. 5. The edges of the matching are indicated by darker lines. This matching is also maximal and complete. It corresponds to the deadlock state represented in Fig. 4.

The above definitions are used to establish an approach that can determine if a given set of nodes is a deadlock set. We begin by the intuitively obvious result that a subgraph generated by a deadlock set must contain a cycle. (A *cycle* is a directed path of a graph that initiates and terminates at the same node.)

• *Theorem 1*
The subgraph generated by a deadlock set contains a cycle.

The proof is straightforward. The tasks in each node in a deadlock set are waiting for a buffer within the deadlock set. Construct a directed path where each arc is between the node where a task is residing and the node containing the buffer for which it is waiting. Since the deadlock set is a finite set of nodes, the directed path must revisit a node, thereby resulting in a cycle. Details of the proof are given in [8, 9].

The above is a necessary condition for a network to be exposed to deadlock. It is now shown that the problem of determining whether a given set X' of nodes is a deadlock set can be transformed into that of the existence of a complete weighted matching of the bipartite graph for the subgraph generated by X'.

• *Theorem 2*
In a routed network $N = (X, A, B, R, M)$, a nonempty connected set $X' \subseteq X$ of nodes is a deadlock set if and only if there exists a complete weighted matching of the bipartite graph $H = (X', R, E)$ for the subgraph generated by X'.

Proof
1. First, we assume that E_0 is a complete weighted matching of H and show that X' is a deadlock set.

We label the edges of E by letting $e_{ijk} \in E_0$ denote the kth edge connecting x_i and r_j, where $1 \leq k \leq \min(b_i, m_j)$. Then the matching E_0 can be associated with the state S such that, for each edge $e_{ijk} \in E_0$, there is a task $t = \langle r_j, k\rangle$ and $S(t) = x_i$. State S is a feasible state since, by definition of a weighted matching for each node $x_i \in X'$ and each route $r_j \in R$, E_0 cannot have more than b_i and m_j edges, respectively.

Consider a task $t = \langle r_j, k\rangle \in T' = \{t \in T | S(t) \in X'\}$ such that $S(t) = x_i$. Then, there is a corresponding edge $e_{ijk} \in E_0$ connecting node x_i and route r_j.

The definition of the bipartite graph for a subgraph implies that the next node after x_i on r_j, say x_q, is in X'. Also, since E_0 is a complete weighted matching, the number of buffers assigned in node x_q, as represented by the number of edges in E_0 connected to x_q, is b_q.

Since this argument holds for an arbitrary task t in T', it holds for all tasks in T'. It follows that S is a deadlock state and X' is a deadlock set.

2. Now it will be shown that if X' is a deadlock set, then H has a complete weighted matching. Let D be the deadlock state that corresponds to the deadlock set X'.

We construct a weighted matching E of H by selecting an edge $e_{ijk} \in E$ for each task $t = \langle r_j, k\rangle$ and $D(t) = x_j$. There are sufficient edges in E to allow such a selection. This is true since H is a bipartite graph for a subgraph.

We must show that E_0 is complete. Consider a node $x_i \in X'$. Since X' is a deadlock set, the number of tasks in $T''' = \{t \in T | D(t) = x_i\}$ is b_i. Also, by construction, there is an edge in E_0 for each task t such that $D(t) = x_i$. Therefore, the number of edges connected to x_i in E_0 is b_i. Since this is true for an arbitrary node $x_i \in X'$, it is true for each node in X'. Hence, E_0 is a complete weighted matching.

This completes the proof.

The next corollary applies the above result to establish that deadlock exposure of a routed network is equivalent to existence of a complete weighted matching of the bipartite graph for some subgraph generated by a nonempty connected set of its nodes. Its proof follows immediately from the definition of deadlock exposure of a routed network and the above theorem.

• *Corollary 1*

A routed network $N = (X, A, B, R, M)$ is exposed to deadlock if and only if there exists a nonempty connected set of nodes $X' \subseteq X$ such that the bipartite graph $H = (X', R, E)$ for the subgraph $S_g = (X', A')$ generated by X' has a complete weighted matching.

The above corollary establishes both necessary and sufficient conditions for a routed network to be exposed to deadlock. It is of interest to note that a set X' of nodes cannot be a deadlock set if the total number of edges in the bipartite graph for the subgraph generated by X' is less than the total number of buffers in the nodes in X'. This is a strong necessary condition that can be easily evaluated.

• *Theorem 3*

Let $H = (X', R, E)$ be the bipartite graph for a subgraph $S_g = (X', A')$ generated by a nonempty connected set X' of nodes. Then, X' is not a deadlock set if the number of edges in E is less than the sum of the number of buffers in X'.

This theorem can be proved by showing its contrapositive, that is, if X' is a deadlock set, then

$$|E| \geq \sum_{x_i \in X'} b_i.$$

By Theorem 2, H has a complete weighted matching, say E_0. So

$$|E| \geq |E_0| \geq \sum_{x_i \in X'} b_i.$$

4. Application of network flow theory

The above results provide a transformation of the problem of determining whether a routed network is exposed to deadlock to that of the existence of a complete weighted matching for the bipartite graph of some subgraph of the routed network. We first develop an approach to provide a maximum weighted matching.

Edmonds [10] has developed a useful algorithm for solving the more general case of the problem of obtaining a maximum weighted matching of a graph. However, for a bipartite graph, it can be readily shown [11] that the matching problem is equivalent to the maximum flow problem in a capacitated network, treated by Ford and Fulkerson [12]. The matching problem of a bipartite graph can be transformed to a network flow problem as follows.

Let $H = (X', R, E)$ be a bipartite graph for a subgraph $S_g = (X', A')$. Define a flow network by adding a "source node," s, and a "sink node," t, to the set $X' \cup R$. The set of arcs for the flow network is obtained by including an arc $(x_i r_j)$ from node x_i to route r_j if there is an edge in E connecting x_i and r_j. To this add a set of arcs from source node s to each node in X', and a set of arcs from each route in R to sink node t. Each arc from s to some node $x \in X'$ will have a flow capacity equal to the number of buffers in x. Each arc $(x_i r_j)$ from node x_i to route r_j can have any finite capacity that is greater than the number of buffers (b_i) in x_i or number of messages (m_j) allowed on route r_j. Finally, each arc from a route $r_j \in R$ to sink node t will have a flow capacity equal to the maximum number of messages allowed on the route r_j.

In such a flow network, any flow that conserves arc flows at each of its nodes is called a "compatible flow." Consider a compatible flow from s to t. The maximum flow leaving each node $x_i \in X'$ is restricted to b_i, and that arriving at each route $r_j \in R$ is restricted to m_j. Also, a weighted matching E_0 of H has no more than b_i edges connected to node x_i and no more than m_j edges connected to route r_j. Furthermore, there is an arc $(x_i r_j)$ in the flow network only if there is a nonempty set E_{ij} of edges in E that connects x_i and r_j. Thus, the maximum flow from s to t in the flow network gives the maximum weighted matching of the bipartite graph. Such a matching is obtained by selecting, for each pair of connected nodes (x_i, r_j) in H, the number of edges equal to the maximum flow obtained for arc $(x_i r_j)$ in the flow network. Such a flow network is defined next.

Let $H = (X', R, E)$ be the bipartite graph for a subgraph $S_g = (X', A')$ generated by X' on a network $N = (X, A, B, R, M)$. Then a *flow network* H_n for H is denoted by $H_n = (V, \alpha, f)$, where V is the set of nodes $X' \cup R \cup \{s, t\}$, and where s is a new source node and t is a new sink node; α, the set of arcs in H, is given by

$$\{(sx_i) | x_i \in X'\} \cup \{(x_i r_j) | x_i \text{ is adjacent to } r_j \text{ in } H\}$$
$$\cup \{(r_j t) | r_j \in R\},$$

and f is a function on α to the positive integers such that

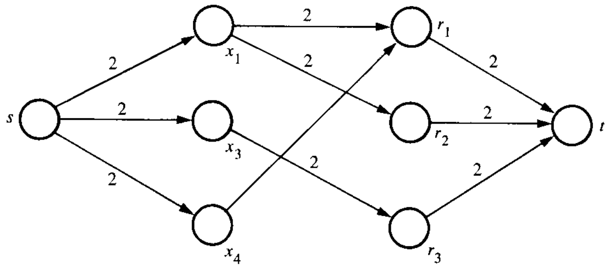


Figure 6 Flow network for the bipartite graph of Fig. 5.

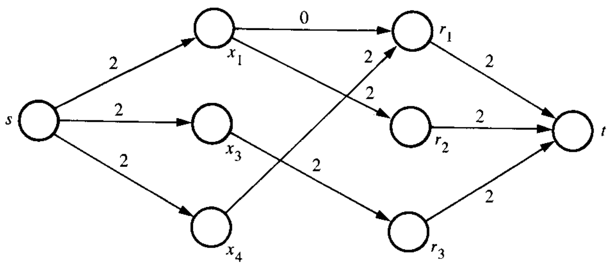


Figure 7 A maximum compatible flow for the flow network of Fig. 6.

1. For each arc $(sx_i) \in \alpha$,
   $f(sx_i) = b_i$.
2. For each arc $(x_ir_j) \in \alpha$,
   $f(x_ir_j)$ = any positive integer $\geq \min(b_i, m_j)$.
3. For each arc $(r_jt) \in \alpha$,
   $f(r_jt) = m_j$.

Here $f(\alpha_i)$ is called the flow capacity of arc $\alpha_i \in \alpha$.

An example of a flow network for the bipartite graph of Fig. 5 is illustrated in Fig. 6. A maximum compatible flow of this flow network is shown in Fig. 7. The maximum weighted matching for the bipartite graph, obtained from this maximum flow, is shown in Fig. 5; it is also a complete weighted matching.

Ford and Fulkerson [12] have described an algorithm to obtain a maximum compatible flow of a given flow network. The complexity of their algorithm has been improved by Dinic [13]. Dinic's algorithm is bounded by kv^2c operations, where v is the number of nodes, c is the number of arcs in the flow network, and k is a constant that is independent of the flow network.

We can now present an approach to determine whether a given routed network $N = (X, A, B, R, M)$ is exposed to deadlock. The approach is based on the above results and is described in four steps.

Step 1 Create an adjacency matrix for the routed network graph.

Step 2 (a) Let the number of nodes in the routed network be n. Then, generate a new n-bit binary number every time this step is entered (from 1 to $2^n - 1$). Proceed to step 2(b). When no new numbers can be generated, the procedure terminates and the routed network is not exposed to deadlock.
(b) For a given binary number β from (a), let X' be the set of nodes that have the corresponding bit on in β. If X' is connected, then generate a subgraph $S_g = (X', A')$ on the network graph $G = (X, A, B)$. Next, define a bipartite graph $H = (X', R, E)$ for S_g as described in Section 2.

Step 3 Check whether

$$|E| \leq \sum_{x_i \in X'} b_i.$$

If so, then by Theorem 2, X' is not a deadlock set. So proceed to step 2 to obtain a new set of nodes. Otherwise, go to step 4.

Step 4 Define a flow network $H_n = (V, \alpha, f)$ for the bipartite graph as described earlier. Obtain a maximum compatible flow in H_n by applying Dinic's algorithm [13], assuming zero initial flow in all arcs in α. Obtain the maximum weighted matching in $H = (X', R, E)$ corresponding to the maximum flow. If the matching is complete, then the routed network is exposed to deadlock and the procedure terminates (Corollary 1). If the matching is not complete, proceed to step 2.

It can be readily seen that, for an algorithm based on the above approach, the total number of operations is dominated by the number of executions of the maximum flow algorithm. For an n-node network, there cannot be more than 2^n subgraphs. So, the total number of operations of such an algorithm will be $O[2^n(n + r)^2c]$, where r is the number of routes and c is the number of arcs of the network graph.

5. Summary

In this paper, we have developed an approach to determine whether a given routed network is exposed to deadlock. Furthermore, this research has established a relation among the deadlock exposure of routed networks, the matching problem in graph theory, and the network flow problem in mathematical programming. These results are applicable to store and forward networks with fixed routing. Further work is needed to extend these results to the general class of operating systems and to more than one class of resources.

Acknowledgments

The research reported here was conducted under the excellent guidance and direction of Dr. V. L. Wallace at the Department of Computer Science, University of North Carolina, Chapel Hill. We wish to express our deep gratitude to Dr. Wallace for his extensive contributions in this effort. (Dr. Wallace is currently at the University of Kansas, Lawrence.)

References and note

1. E. G. Coffman, M. J. Elphick, and A. Shoshani, "System Deadlocks," *Computing Surv.* **3,** 67–78 (1971).
2. A. N. Habermann, "Prevention of System Deadlocks," *Commun. ACM* **12,** 373-377, 385 (1969).
3. R. C. Holt, "On Deadlocks in Computer Systems," Ph.D. Thesis, Department of Computer Science, Cornell University, Ithaca, NY, 1971.
4. Some of the results described here were presented at the 1977 ACM Computer Science Conference.
5. R. E. Kahn and W. R. Crowther, "Flow Control in Resource Sharing Computer Networks," *IEEE Trans. Commun.* **COM-20,** 539–545 (1972).
6. L. Kleinrock, *Queuing Systems Volume 2: Computer Applications*, John Wiley & Sons, Inc., New York, 1976, pp. 439–440.
7. E. Raubold and J. Haenle, "A Method of Deadlock-free Resource Allocation and Flow Control in Packet Networks," *Proceedings of the Third International Conference on Computer Communication* (Toronto, August 1976), p. 483-487.
8. V. Ahuja, "Exposure of Routed Networks to Deadlock," Ph.D. Thesis, Department of Computer Science, University of North Carolina, Chapel Hill, NC, 1976.
9. V. Ahuja, "An Algorithm to Check Network States for Deadlock," *IBM J. Res. Develop.* **23,** 82-86 (1979).
10. J. Edmonds, "Paths, Trees and Flowers," *Can. J. Math.* **17,** 449–467 (1965).
11. R. G. Busacker and T. L. Saaty, *Finite Graphs and Networks: An Introduction with Applications*, McGraw-Hill Book Co., Inc., New York, 1965.
12. L. R. Ford, Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.
13. E. A. Dinic, "Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation," *Soviet Math. Dokl.* **11,** 1277-1280 (1970).

Received February 16, 1979; revised August 8, 1979

The author is located at the IBM System Communications Division laboratory, Research Triangle Park, North Carolina 27709.