Requirements for Deadlock-Free, Adaptive Packet Routing

Robert Cypher*

Luis Gravano*†

*IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

†CRAAG
IBM Argentina
Buenos Aires, Argentina

Abstract

This paper studies the problem of deadlock-free packet routing in parallel and distributed architectures. We present three main results. First, we show that the standard technique of ordering the queues so that every packet always has the possibility of moving to a higher ordered queue is not necessary for deadlock-freedom. Second, we show that every deadlock-free, adaptive packet routing algorithm can be restricted, by limiting the adaptivity available, to obtain an oblivious algorithm which is also deadlock-free. Third, we show that any packet routing algorithm for a cycle or torus network which is free of deadlock and which uses only minimal length paths must require at least three queues in some node. This matches the known upper bound of three queues per node for deadlock-free, minimal packet routing on cycle and torus networks.

1 Introduction

This paper studies the problem of deadlock-free packet routing in parallel and distributed architectures. A wide range of packet routing algorithms with differing properties and costs have been proposed [1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 14, 15, 16, 17, 19, 20]. In this paper we will focus on a particularly simple and important class of routing algorithms which we will call *queue-reservation* algorithms. A queue-reservation algorithm consists of rules that specify to which queues a packet may move based solely on the queue currently holding the packet, the packet's source node, and the packet's destination node. A packet is allowed to move from its current queue to any other queue at any time, provided that the other queue is empty and that the move is allowed by the routing algorithm. Queue-reservation algorithms can be implemented efficiently in hardware as they require only local information to make routing decisions, they are inherently asynchronous and therefore do not require a global clock, and they do not require the creation or exchange of any special packets containing only control information. Furthermore, adaptive queue-reservation algorithms allow packets to avoid congestion, thus permitting high throughput in the network. As a result of these advantages, queue-reservation algorithms have been widely studied and implemented.

The primary disadvantage of queue-reservation techniques is that they require that each node contain some minimum number of queues. Although a great deal of research has been devoted to the problem of minimizing the storage requirements of queue-reservation algorithms [2, 4, 5, 6, 8, 10, 12, 15, 17, 18, 19, 20], very little is known in terms of lower bounds on the storage which is required by such algorithms. Our goal in this paper is to characterize the properties which these algorithms must have in order to be free of deadlock and to use these properties to prove lower bounds on storage requirements.

One well-known technique for proving freedom from deadlock is to order the queues so that every packet always has the possibility of moving to a higher ordered queue [8]. Providing such an ordering of the queues is the standard technique for proving freedom from deadlock and has been used by many researchers [2, 4, 5, 6, 8, 10, 12, 15, 17, 19, 20]. Therefore, it seems plausible that the existence of such an ordering of the queues is a necessary condition for freedom from deadlock. In fact, in the special case of oblivious queue-reservation algorithms, Toueg and Steiglitz have shown that the existence of such an ordering of the queues *is* necessary for deadlock-freedom [18]. However, in this paper we will present an adaptive queue-reservation algorithm which is provably free of deadlock and for which no ordering of the queues can be defined such that every packet always has the possibility of moving to a higher ordered queue. Thus, in the case of adaptive routing the technique of ordering the queues is sufficient but not necessary for avoiding deadlock.

On the other hand, we will prove that every deadlock-free, adaptive queue-reservation algorithm can be restricted, by limiting the adaptivity available, to obtain an oblivious algorithm which is also deadlock-free. As a result, we will be able to use lower bounds on the storage requirements of oblivious routing algorithms to obtain lower bounds on the storage requirements of adaptive routing algorithms. In particular, we will show that any adaptive queue-reservation algorithm for a cycle or torus network which is free of deadlock and which uses only minimal length paths must require at least three queues in some node. This matches the known upper bound of three queues per node for deadlock-free, minimal routing on cycle [7] and torus networks [2].

The remainder of this paper is organized as follows. Definitions and a formal description of the routing model are given in Section 2. Section 3 presents an example of a deadlock-free, adaptive routing algorithm in which it is impossible to order the queues so that every packet always has the possibility of moving to a higher ordered queue. The fact that every deadlock-free, adaptive queue-reservation algorithm can be restricted to obtain a deadlock-free, oblivious algorithm is proven in Section 4. Lower bounds on the storage requirements for deadlock-free minimal queue-reservation algorithms are given in Section 5.

2 Preliminaries

We will view a routing network as being an undirected graph in which the nodes represent processors and the edges represent communication links. Each node contains a set of queues, one of which will be called an *injection queue*, another one of which will be called a *delivery queue*, and the remainder of which will be called *standard queues*. Packets can enter the routing network only by being placed in an empty injection queue in their source node, and they can be removed from the network only when they are in the delivery queue of their destination node. We will assume throughout that each queue can hold exactly one packet and that the number of queues is finite.

Given the queue in which a packet is currently stored, and given the packet's source and destination nodes, a *routing algorithm* specifies a set of queues to which the packet may be moved. More formally, the *color* of a packet is the pair (s, d) where s is the packet's source node and d is the packet's destination node. We will say that a queue has color c if it contains a packet with color c. A routing algorithm A is a function which associates a set of queues, called a *waiting set*, with each possible queue and color pair (q, c). The waiting set which A associates with the pair (q, c) will be denoted $A(q, c)$. Given a routing algorithm A, a queue q is *reachable* by a packet p with color c if and only if there exists some path $q_0, q_1, \ldots, q_k$ such that q_0 is the injection queue in p's source node, $q_k = q$, and for all i, $1 \leq i \leq k$, $q_i \in A(q_{i-1}, c)$. It is required that the waiting set $A(q, c)$ be empty if and only if either q is a delivery queue or q is not reachable by a packet with color c.

All of the queues in a waiting set $A(q, c)$ must either be in the node which contains q or in neighboring nodes (that is, nodes that are connected by an edge to the node containing q). An injection queue is never allowed to appear in a waiting set, and a delivery queue must only be reachable by packets destined for the node containing the delivery queue. Furthermore, if $q_2 \in A(q_1, c)$

and if either q_1 is an injection queue or q_2 is a delivery queue, then q_1 and q_2 must be in the same node. Thus injection and delivery queues are used only for placing new packets in the network and for removing packets once they have reached their destination [1].

The routing algorithm operates asynchronously. A packet with color c may move from a queue q_1 to any empty queue $q_2 \in A(q_1, c)$ at any time, and a new packet with an arbitrary destination may be placed in an empty injection queue at any time. Packets may be transmitted in either store-and-forward [15] or virtual cut-through [11] mode. The only requirement is that when a packet is moved from one queue to another, it occupies both of the queues for a finite amount of time, and after a finite amount of time the former queue becomes empty.

A routing algorithm is *oblivious* if every waiting set contains at most one queue, and it is *adaptive* otherwise. Routing algorithm A is a *restriction* of routing algorithm B if and only if for every pair (q, c), $A(q, c) \subseteq B(q, c)$, and for some pair (q, c), $A(q, c) \neq B(q, c)$. A routing algorithm is *minimal* if every packet is routed from its source node to its destination node while visiting the minimum number of nodes possible. Note that the concept of minimality is based on the number of nodes visited, rather than the number of queues visited.

A *configuration* is a nonempty set S of queues such that each queue q in S is either empty or has color c where q is reachable by a packet with color c. The set of queues S will be called the *critical set* of the configuration. Given a configuration T with critical set S and given any queue $q \in S$, the notation $T(q)$ will denote q's color in configuration T (or the value "empty" if it does not contain a packet). A *deadlock configuration* for a routing algorithm A is a configuration with a critical set S such that none of the queues in S is a delivery queue, none of the queues in S is empty, and for each queue q in S, q has color c where $A(q, c) \subseteq S$. A configuration is *routable* if and only if it is possible to start with an empty network and to route packets so as to obtain the configuration.

[1] It should be noted that the injection and delivery queues are introduced only to simplify the description of the model, and that they need not be physically present in an actual routing network.

A routing algorithm is *deadlock-free* if and only if it has no routable deadlock configuration. It is straightforward to verify that this definition of deadlock-freedom does in fact correspond to the impossibility of obtaining deadlock when using the given routing algorithm. Finally, given any two configurations T' and T'' with critical sets S' and S'', respectively, $T' \oplus T''$ will denote the configuration T with critical set $S = S' \cup S''$ in which for each queue $q \in S'$, $T(q) = T'(q)$, and for each queue $q \in S'' \setminus S'$, $T(q) = T''(q)$. Thus $T' \oplus T''$ is obtained by taking configuration T' and adding to it all of those queues in T'' which are not also in T'.

3 Deadlock-Freedom Without Ordering Queues

In this section we will show that the standard technique of ordering the queues so that every packet always has the possibility of moving to a higher ordered queue is not necessary for the prevention of deadlock. In particular, we will give a simple example of an adaptive routing algorithm which is provably free of deadlock and yet has no such ordering of the queues.

The example is routing algorithm A shown in Figure 1, in which each circle represents a queue and each arc represents a possible move between queues. There are three injection queues labeled I_1, I_2 and I_3, and three delivery queues labeled D_1, D_2 and D_3. In addition, there are six standard queues labeled X_1, X_2, X_3, Y_1, Y_2 and Y_3. We will consider only three colors of packets, namely C_1, C_2 and C_3, where packets with color C_i, $1 \leq i \leq 3$, are injected in queue I_i and delivered from queue D_i. The label associated with each arc specifies which color packets are allowed to make the given move between queues. For example, $A(I_1, C_1) = \{X_1\}$, $A(X_1, C_1) = \{X_2, Y_1\}$, and $A(X_1, C_2) = \{X_3\}$. Of course a complete routing algorithm would provide routes for packets with other colors and would include an assignment of the queues to the nodes in a routing network. However, it is straightforward to extend the given example by adding additional queues for the packets with other colors and to assign the queues to nodes in a routing network without changing the deadlock or queue ordering properties of the example.

Lemma 3.1 *The routing algorithm A shown in Figure 1 is free of deadlock.*

Proof: Assume for the sake of contradiction that deadlock is possible, in which case there must be some routable deadlock configuration with a nonempty critical set S. Clearly, the delivery queues cannot appear in S. Similarly, Y_1, Y_2 and Y_3 cannot appear in S because they are only reachable by packets which are able to move directly to a delivery queue. Also, note that if injection queue I_i, $1 \leq i \leq 3$, is in S, then queue X_i must also be in S. Therefore, at least one of the queues X_i must be in S. Because none of the queues Y_i is in S, it follows that if X_1 is in S it must have color C_2 in the deadlock configuration, if X_2 is in S it must have color C_1 in the deadlock configuration, and if X_3 is in S it must have color C_3 in the deadlock configuration. Note that X_3 must be in S, because otherwise either X_1 or X_2 must be in S, and $X_3 \in A(X_1, C_2)$ and $X_3 \in A(X_2, C_1)$. Because $X_1 \in A(X_3, C_3)$ and $X_2 \in A(X_3, C_3)$, both X_1 and X_2 must be in S. Therefore, the deadlock configuration must include a C_2 packet in X_1 and a C_1 packet in X_2. However, it is impossible to simultaneously route a C_2 packet to X_1 and a C_1 packet to X_2, so the deadlock configuration is not routable, which is a contradiction. □

Lemma 3.2 *There is no ordering of the queues shown in Figure 1 such that every packet always has the possibility of moving to a higher ordered queue.*

Proof: Assume for the sake of contradiction that such an ordering is possible. Because $A(X_1, C_2) = \{X_3\}$ and $A(X_2, C_1) = \{X_3\}$, queue X_3 must be higher ordered than both X_1 and X_2. However, $A(X_3, C_3) = \{X_1, X_2\}$, so either X_1 or X_2 (or both) must be higher ordered than X_3, which is a contradiction. □

Combining the two previous lemmas yields the following theorem.

Theorem 3.3 *There exists an adaptive routing algorithm which is free of deadlock, and for which there is no ordering of the queues such that every packet always has the possibility of moving to a higher ordered queue.*

4 Restrictions of Adaptive Routing Algorithms

In this section we will show that every deadlock-free, adaptive packet routing algorithm can be restricted to obtain an oblivious algorithm which is also deadlock-free. The proof will depend on the following lemma.

Lemma 4.1 *Let A be any deadlock-free, adaptive routing algorithm, let q_1 be any queue, and let c_1 be any color such that $|A(q_1, c_1)| \geq 2$. Let q_2 be any queue such that $q_2 \in A(q_1, c_1)$, and let B be the restriction of A obtained by removing q_2 from the waiting set associated with (q_1, c_1). If B is subject to deadlock, then there must exist some routable deadlock configuration for B in which queue q_1 contains a packet with color c_1.*

Proof: Because B is subject to deadlock, there must exist some configuration T which is a deadlock configuration for B and which is routable by B. Because B is a restriction of A, it follows that configuration T is also routable by A. However, A is deadlock-free, so T must not be a deadlock configuration for A. Therefore, queue q_1 must have color c_1 in configuration T. □

Theorem 4.2 *Given any adaptive, deadlock-free routing algorithm A, there exists an oblivious, deadlock-free routing algorithm B which is a restriction of A.*

Proof Sketch: Assume for the sake of contradiction that the claim is false. Then there must exist some adaptive, deadlock-free routing algorithm A such that every routing algorithm A' which is a restriction of A is subject to deadlock. Let A be such a deadlock-free routing algorithm, let q_1 be any queue, and let c_1 be any color such that $|A(q_1, c_1)| \geq 2$. Let q_2 and q_3 be any distinct queues such that $q_2 \in A(q_1, c_1)$ and $q_3 \in A(q_1, c_1)$, let A' be the restriction of A obtained by removing q_2

from the waiting set associated with (q_1, c_1), and let A'' be the restriction of A obtained by removing q_3 from the waiting set associated with (q_1, c_1). It follows from Lemma 4.1 that there exists a configuration T' (similarly, T'') which is a routable deadlock configuration for A' (similarly, A'') and in which queue q_1 contains a packet with color c_1. Let $T = T' \oplus T''$ and let S be the critical set of T. Let R be the configuration which also has critical set S but in which all of the queues are empty. Note that the following properties hold:

Property 1: Configuration T is a deadlock configuration for A.

Property 2: Configuration R is a routable configuration for A.

Property 3: The set S is the critical set of both configuration T and configuration R.

Property 4: Every nonempty queue q in R has a color c such that $A(q, c) \subseteq S$.

We will define an algorithm for transforming R and T while maintaining Properties 1 through 4 listed above. The algorithm will repeatedly add packets to empty queues in R until none of the queues in R is empty. At this point R will be a routable deadlock configuration, which will be the desired contradiction. The algorithm for transforming R and T consists of repeatedly performing the following subroutine until R contains no empty queues.

First, select an arbitrary queue q which is empty in R. Let $c = T(q)$. Because queue q is reachable by some packet p with color c (from the definition of a configuration), there must exist a simple path from p's injection queue to queue q. Define the configuration P in which the critical set consists of all of those queues that appear in this simple path, and in which all of the queues in the critical set contain a packet with color c. Transform R to become the configuration obtained by adding P and R (that is, perform the assignment $R \leftarrow P \oplus R$), transform T to become the configuration obtained by adding P and T (that is, perform the assignment $T \leftarrow P \oplus T$), and let S be the critical set of the transformed configurations R and T. Note that at this point R is routable, because it is possible to first route packets with the desired colors to all of the nonempty queues in R which are not in P and then fill the queues in P with packets with color c. Also, note that at this point T may not be a deadlock configuration, because it is possible that some of the packets in P have waiting sets that include queues which are not in S.

Next, select an arbitrary queue q' in the simple path described above such that $A(q', c) \not\subseteq S$ (if such a queue exists). Let q'' be the successor of q' in the simple path described above (note that q'' must exist if q' exists, because $A(q, c) \subseteq S$ so $q' \neq q$). Let A' be the restriction of A obtained by removing q'' from the waiting set associated with (q', c). It follows from Lemma 4.1 that there exists a configuration D which is a routable deadlock configuration for A' and in which queue q' contains a packet with color c. Let D' be the configuration with the same critical set as D but in which all of the queues are empty. Transform R to become the configuration obtained by adding R and D' (that is, perform the assignment $R \leftarrow R \oplus D'$), transform T to become the configuration obtained by adding T and D (that is, perform the assignment $T \leftarrow T \oplus D$), and let S be the critical set of the transformed configurations R and T. Repeat this procedure of selecting a queue q' in the simple path such that $A(q', c) \not\subseteq S$ and transforming R, T and S until no such queue q' exists. When no such queue q' exists, return from the subroutine.

Note that upon returning from the subroutine Properties 1 through 4 above must hold. Also, note that any queue in R which was nonempty before calling the subroutine will again be nonempty after calling the subroutine. Finally, note that following the call to the subroutine, R contains at least one additional nonempty queue. Because the number of queues is finite, this procedure must terminate, at which point R is both routable by A and a deadlock configuration for A, which is a contradiction. □

5 Minimal Routing in Cycle and Torus Networks

In this section we will prove lower bounds on the number of queues per node that are required for deadlock-free, minimal routing in cycle and torus networks. Our approach will be to first prove a lower bound on the queue requirements of deadlock-free, minimal, oblivious routing algorithms for cycle networks. We will then use this lower bound, along with Theorem 4.2 and the fact that a torus network can be decomposed into disjoint cycles, to obtain a lower bound on the queue requirements of deadlock-free, minimal routing algorithms for both cycle and torus networks.

Lemma 5.1 *Let routing algorithm A be any deadlock-free, minimal, oblivious routing algorithm for a cycle network with n nodes. The cycle network must contain at least $3n - 12$ standard queues.*

Proof: Because A is deadlock-free and oblivious, it follows that there exists an ordering of the queues such that every packet visits the queues in ascending order [18]. Let $k = \lfloor n/2 \rfloor - 1$ (so either $n = 2k+2$ or $n = 2k+3$). We will say that a packet is routed in the *clockwise* direction if it visits queues in nodes of the form $i, (i+1) \bmod n, (i+2) \bmod n, \ldots, j,$ and in the *counterclockwise* direction otherwise. Note that for each node i, $0 \le i < n$, algorithm A routes packets from node i to node $(i+k) \bmod n$ in the clockwise direction. Therefore, for each node i, $0 \le i < n$, there must exist an ascending sequence of standard queues $s_{i,i}, s_{i,(i+1)\bmod n}, \ldots, s_{i,(i+k)\bmod n}$ where each queue of the form $s_{i,j}$ is located in node j (for example, let $s_{i,j}$ be the highest ordered standard queue in node j which is visited by a packet with source node i and destination node $(i+k) \bmod n$). For each i, $0 \le i < n$, let $S_i = s_{i,i}, s_{i,(i+1)\bmod n}, \ldots, s_{i,(i+k)\bmod n}$ denote the ascending sequence of standard queues beginning in node i. Let $h = n-1-k$, note that $S_h = s_{h,h}, s_{h,h+1}, \ldots, s_{h,n-1}$, and note that S_0 and S_h are disjoint.

We will say that a sequence of queues is a *clockwise-increasing* (similarly, *counterclockwise-increasing*) sequence if when the queues are visited in ascending order, the nodes containing the queues are visited in clockwise (counterclockwise) order. We will show that there must exist at most three mutually disjoint clockwise-increasing sequences of queues, the total length of which is at least $n + k$. There are two cases:

Case 1: There exists a clockwise-increasing sequence of standard queues $X = x_0, x_1, \ldots, x_{n-1}$ such that for each i, $0 \le i < n$, x_i is located in node i. In this case, we have two subcases:

- **Case 1a:** There exists an a, $0 \le a \le n-1$, such that S_a and X are disjoint. In this case, the two disjoint clockwise-increasing sequences are S_a and X, and their total length is $n+k+1$.

- **Case 1b:** For each i, $0 \le i \le n-1$, S_i and X intersect. In this case, let a be the largest value of i, $0 \le i \le n-1$, such that there exists a value $i' \ge i$ where $s_{i,i'} = x_{i'}$. Let a' be any value such that $a' \ge a$ and $s_{a,a'} = x_{a'}$. Let $b = (a+1) \bmod n$ and let b' be any value such that $s_{b,b'} = x_{b'}$. Note that $a' \ge a \ge k \ge b'$. Let Y be the sequence

$$s_{b,b}, \ldots, s_{b,b'-1}, x_{b'}, \ldots, x_{a'},$$
$$s_{a,a'+1}, \ldots, s_{a,(a+k)\bmod n}.$$

  The sequence Y is clockwise-increasing and has length $n+k$.

Case 2: There does not exist such a sequence X. In this case, we have two subcases:

- **Case 2a:** There exists an a, $0 \le a \le h$, such that S_a and S_0 are disjoint and such that S_a and S_h are disjoint. In this case, the three disjoint clockwise-increasing sequences are S_0, S_a, and S_h, and their total length is $3k+3 \ge n+k$.

- **Case 2b:** For each i, $0 \le i \le h$, either S_i and S_0 intersect or S_i and S_h intersect, but not both. In this case, let a be the largest value of i in the range $0 \le i \le h$ such that S_i and S_0 intersect. Let a' be any value such that $s_{a,a'} = s_{0,a'}$. Let $b = a + 1$ and let b' be any value such that $s_{b,b'} = s_{h,b'}$ (note that such a b' must exist because of the definition of a and the fact that S_h does not intersect S_0). Let Y be the sequence

$$s_{0,0}\,,\ \ldots\,,\ s_{0,a'-1}\,,\ s_{a,a'}\,,\ \ldots\,,\ s_{a,a+k}$$

and let Z be the sequence

$$s_{b,b}\,,\ \ldots\,,\ s_{b,b'}\,,\ s_{h,b'+1}\,,\ \ldots\,,\ s_{h,n-1}.$$

Note that Y and Z are clockwise-increasing sequences and that they must be disjoint (because otherwise there would exist a clockwise-increasing sequence X spanning all of the nodes). Also, note that the length of Y is $a+k+1$ and the length of Z is $n-a-1$, so their total length is $n+k$.

Thus, in any case there must exist at most three mutually disjoint clockwise-increasing sequences of queues, the total length of which is at least $n+k$. An analogous argument can be used to show that there must exist at most three mutually disjoint counterclockwise-increasing sequences of queues, the total length of which is at least $n+k$. Because a clockwise-increasing sequence of queues and a counterclockwise-increasing sequence of queues can intersect in at most one queue, it follows that the entire collection of clockwise-increasing sequences and counterclockwise-increasing sequences contains at least $(n+k)+(n+k)-(3*3) = 2n+2k-9 \ge 3n-12$ distinct queues. □

Theorem 5.2
Let routing algorithm A be any deadlock-free, minimal routing algorithm for a cycle network with 13 *or more nodes or for a torus network in which at least one of the dimensions is of length* 13 *or greater. The cycle or torus network must contain at least one node which has three or more standard queues.*

Proof: The claim for a cycle network follows immediately from Theorem 4.2 and Lemma 5.1. The claim for a torus network follows from Theorem 4.2, Lemma 5.1, and the observation that a torus can be decomposed into disjoint cycles such that all minimal length paths between pairs of nodes within a cycle lie within the cycle. □

References

[1] Baruch Awerbuch, Shay Kutten, and David Peleg. Efficient deadlock-free routing. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 177–188, 1991.

[2] Robert Cypher and Luis Gravano. Adaptive, deadlock-free packet routing in torus networks with minimal storage. Technical Report RJ 8571, IBM Almaden Research Center, January 1992. Also to appear in *Proc. 1992 Intl. Conf. on Parallel Processing.*

[3] W. J. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36:547–553, May 1987.

[4] Sergio A. Felperin, Hernán Laffitte, Guillermo Buranits, and Jorge L.C. Sanz. Deadlock-free minimal packet routing in the torus network. Technical Report TR:91-22, IBM Argentina, CRAAG, 1991.

[5] B. Gavish, P.M. Merlin, and P.J. Schweitzer. Minimal buffer requirements for avoiding store-and-forward deadlock. Technical Report RC 6672, IBM T.J. Watson Research Center, August 1977.

[6] Inder S. Gopal. Prevention of store-and-forward deadlock in computer networks. *IEEE Transactions on Communications*, 33(12):1258–1264, December 1985.

[7] Luis Gravano, Gustavo D. Pifarré, Sergio A. Felperin, and Jorge L.C. Sanz. Adaptive deadlock-free worm-hole routing with all minimal paths. Technical Report TR:91-21, IBM Argentina, CRAAG, 1991.

[8] K.D. Günther. Prevention of deadlocks in packet-switched data transport systems. *IEEE Transactions on Communications*, 29(4), April 1981.

[9] Peter A.J. Hilbers and Johan J. Lukkien. Deadlock-free message routing in multicomputer networks. *Distributed Computing*, 3:178–186, 1989.

[10] C.R. Jesshope, P.R. Miller, and J.T. Yantchev. High performance communications in processor networks. In *Proc. 16th Intl. Symposium on Computer Architecture*, pages 150–157, 1989.

[11] P. Kermani and L. Kleinrock. Virtual Cut-Through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[12] S. Konstantinidou. Adaptive, minimal routing in hypercubes. In *Proc. 6th. MIT Conference on Advanced Research in VLSI*, pages 139–153, 1990.

[13] S. Konstantinidou and L. Snyder. The Chaos router: A practical application of randomization in network routing. In *Proc. 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 21–30, 1990.

[14] Yishay Mansour and Boaz Patt-Shamir. Greedy packet scheduling on shortest paths. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 165–175, 1991.

[15] P.M. Merlin and P.J. Schweitzer. Deadlock avoidance in store-and-forward networks. 1: Store-and-forward deadlock. *IEEE Transactions on Communications*, 28(3):345–354, March 1980.

[16] Yoram Ofek and Moti Young. Principles for high speed network control: loss-less and deadlock-freeness, self-routing and a single buffer per link. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 161–175, 1990.

[17] Gustavo D. Pifarré, Luis Gravano, Sergio A. Felperin, and Jorge L.C. Sanz. Fully-adaptive minimal deadlock-free packet routing in hypercubes, meshes, and other networks. In *Proc. 3rd ACM Symp. on Parallel Algorithms and Architectures*, pages 278–290, 1991.

[18] Sam Toueg and Kenneth Steiglitz. Some complexity results in the design of deadlock-free packet switching networks. *SIAM Journal on Computing*, 10(4):702–712, November 1981.

[19] Sam Toueg and Jeffrey D. Ullman. Deadlock-free packet switching networks. *SIAM Journal on Computing*, 10(3):594–611, August 1981.

[20] J. Yantchev and C.R. Jesshope. Adaptive, low latency, deadlock-free packet routing for networks of processors. *IEE Proc., Pt. E*, 136(3):178–186, May 1989.

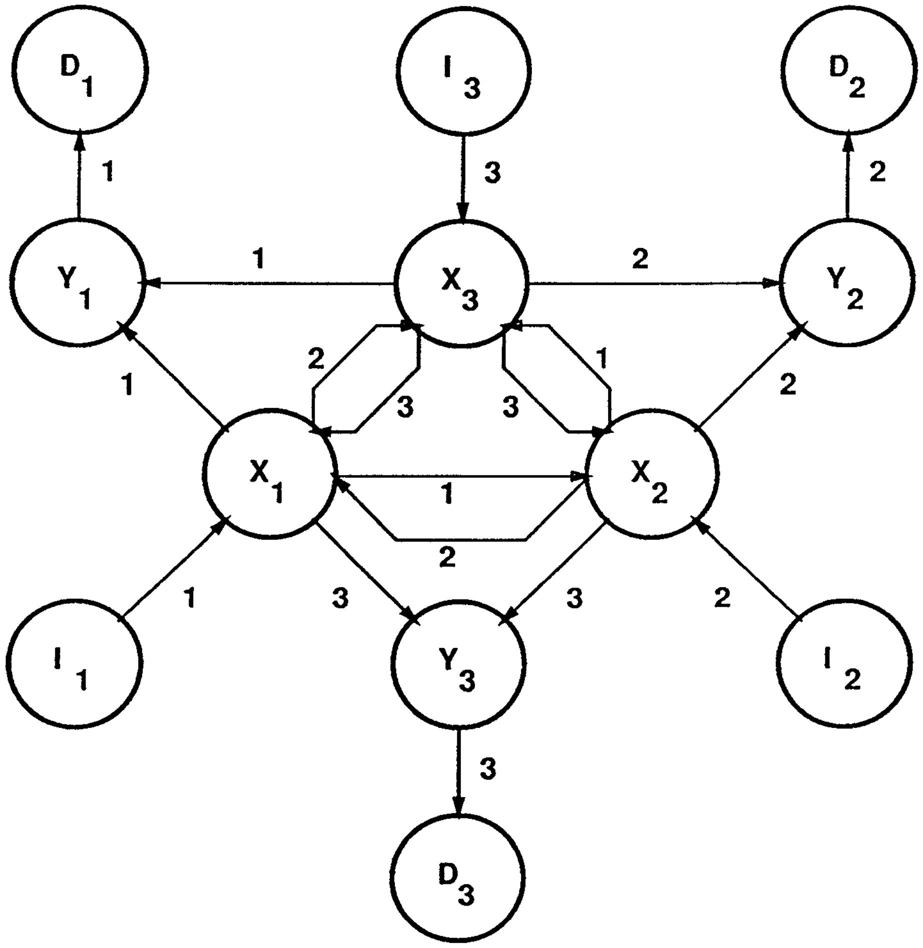

Figure 1: A deadlock-free, adaptive routing algorithm A for which the technique of ordering the queues cannot be used to prove freedom from deadlock.