Deadlock-Free Adaptive Wormhole Routing with Disha Concurrent

Anjan K.V., Timothy Mark Pinkston, Jose Duato

CENG Technical Report 95-21

Department of Electrical Engineering - Systems
University of Southern
Los Angeles, California 90089-2562
(213)740-4482

Deadlock-Free Adaptive Wormhole Routing with Disha Concurrent

Anjan K. V. Timothy Mark Pinkston

Electrical Engineering - Systems Department
University of Southern California
3740 McClintock Avenue, EEB-208
Los Angeles, CA 90089 - 2562
{anjan@truth, tpink@charity}.usc.edu

José Duato

Facultad de Informática
Universidad Politécnica de Valencia
P.O.B. 22012. 46071
Valencia, SPAIN
jduato@aii.upv.es

Abstract:

In previous work, a necessary and sufficient condition for deadlock-free adaptive routing was proposed. This paper generalizes that theory by considering a mixed set of resources (edge and central buffers). Also, in previous work, a simple, efficient and cost effective deadlock recovery strategy called Disha was proposed which is based on the concept of routing deadlocked messages on a deadlock-free lane as opposed to aborting them. As proposed, the scheme requires mutual exclusive access to the deadlock-free lane implemented by a Token. This paper extends the work by applying the generalized theory developed to relax the sequential recovery requirement and allow simultaneous recovery from deadlocks. The proposed extension to Disha does not necessitate any additional resource cost for n-dimensional meshes. For n-dimensional toroids, however, an additional central buffer is required. In both cases, the Token resource is completely eliminated. With this extension, Disha is applicable to any topology with a Hamiltonian path, including k-ary n-cube networks, and tree-based networks.

1.0 Introduction:

Interconnection networks form the backbone for communication in multiprocessor/multi-computer environments. Routing latency and throughput largely determine overall system performance. Communication efficiency can be enhanced by incorporating wormhole switching [19] and adaptive routing [13]. This potent combination is, unfortunately, susceptible to deadlocks.

Avoidance has been the traditional solution to deadlocks. Many avoidance strategies proposed limit the adaptivity of the routing algorithm [5, 7, 18]. Restricting adaptivity curtails performance and may not allow packets to route around faults. Routing adaptivity can be enhanced by increasing the number of virtual channels [9, 17]. However, additional virtual channels increase the complexity of the router crossbar and the Virtual Channel Controller, leading to increased router clock cycles and a consequent cost/performance trade-off [6]. Other avoidance schemes [10] devote virtual channels specifically to prevent deadlocks while allowing fully adaptive routing on others. This also leads to a reduced utilization of existing router resources.

Deadlock recovery as a viable alternative to prevention has only recently begun to gain consideration [2, 14]. Prior research has shown that deadlocks are generally infrequent. Because deadlocks are rare it does not make sense to limit the routing algorithm. This has motivated the development of novel routing strategies based on recovery from deadlocks designed to make the common case fast. Compressionless Routing [14] is one such strategy that detects potential deadlock situations and recovers from them by simply killing the deadlocked packets. *Disha*[†] [2] is a more efficient deadlock recovery scheme which is not based on "regressive" abort-and-retry but, rather, on "progressive" re-direction of deadlocked packets. Figure 1 shows a flow diagram of the *Disha* approach. As shown, *Disha* permits unrestricted routing on all existing virtual channels (edge buffers), and, thus, results in *true* fully adaptive (even non-minimal) routing. Progressive

[†] *Disha* means "*direction*" in *Hindi*.

recovery from deadlocks is through an additional Deadlock Buffer central to the router which is allocated physical channel bandwidth to route deadlocked packets, possibly de-allocating channels temporarily from other packets. Routers designed in this fashion are not only simple but also potentially faster. Exhaustive simulations confirm that this scheme is extremely efficient even when cycles are broken sequentially.

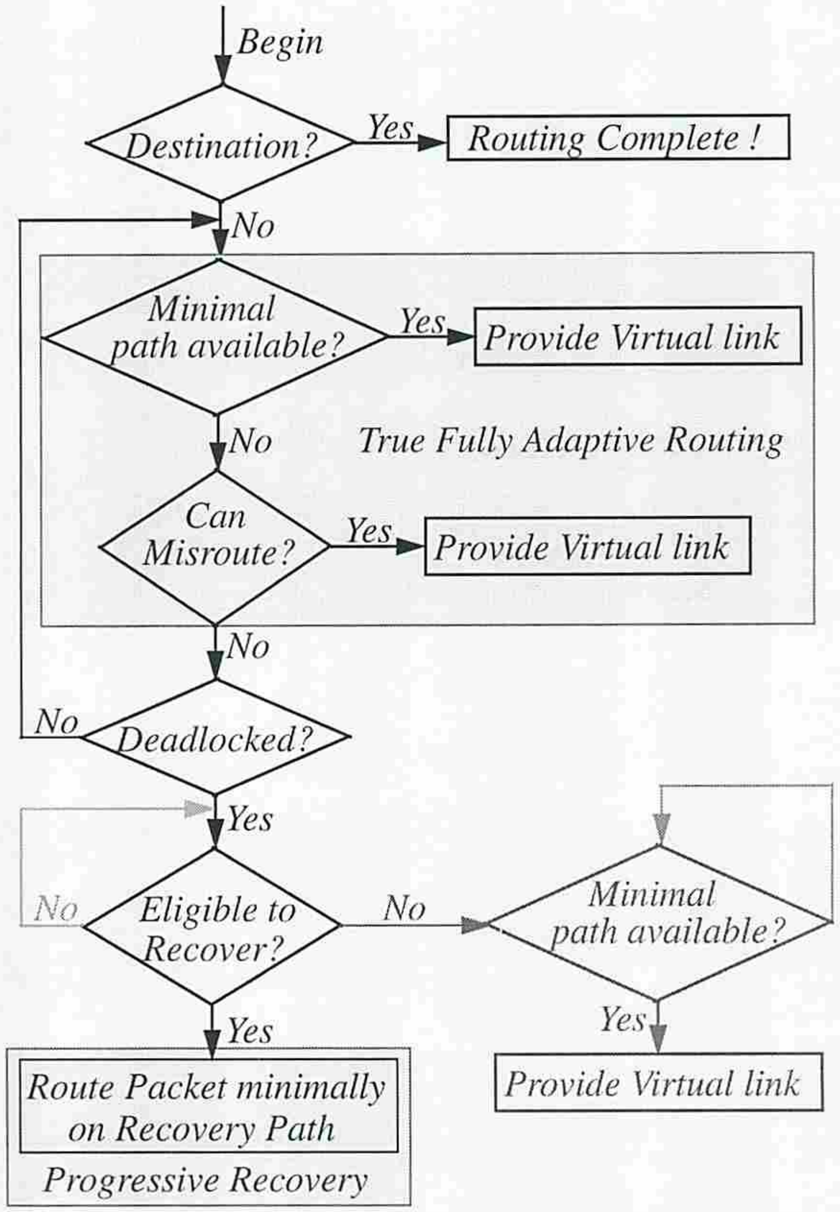


Figure 1: The *Disha* Algorithm.

The green arc is necessary for Sequential recovery to implement mutual exclusion on DBs. The red arc signifies that a once deadlocked packet is now able to proceed (deadlocked cycle broken by another packet in the cycle through the DB path).

Mutual exclusion on the Deadlock Buffers was assumed for recovery in [2]. An asynchronous Token scheme for achieving mutual exclusion is proposed in [1] that drastically reduces propagation time and hides much of the Token capture latency. This scheme is shown to be deadlock-free and efficient, and it confirms the notion that a single Deadlock Buffer central to the router is sufficient to recover from deadlocks. The major drawback of this proposed implementation is the Token logic. The Token forms the basis through which recovery is achieved and, thereby, presents a single point of failure. A less significant drawback is the fact that recovery from cycles is sequential. This could affect performance if deadlocks get clustered in time or become frequent (though this is unlikely for various traffic distributions as simulations have shown).

A solid theory to design and prove that avoidance schemes are deadlock-free has been developed in [10, 11]. In this paper, we generalize the theory to support a mixed set of resources (edge and central buffers) using the notion of buffers as opposed to channels. This paper also extends the work in [1, 2] and presents a method for eliminating the Token logic in *Disha.* For k-ary n-cube meshes, no additional buffers are required to guarantee deadlock freedom. As of now, toroids require an additional central buffer. With this augmentation, *Disha* is no longer susceptible to single-point failures due to Token propagation, capture and release. Moreover, parallel recovery from single deadlock cycles and simultaneous recovery from multiple deadlock cycles is now possible with *Disha*. Using the generalized theory derived, we formally prove that *Disha* with concurrent recovery is deadlock-free. The remainder of this paper is organized as follows. The next section presents a brief background of *Disha*. Section 3 presents our new theory for deadlock freedom based on the notion of buffers that is applicable to both avoidance and recovery schemes. Section 4 applies this theory to *Disha* Concurrent. Section 5 gives performance results and Section 6 concludes with a brief summary and scope for future work.

2.0 Overview of *Disha*:

Disha aims at optimizing performance in the absence of deadlocks. It operates under the most relaxed condition of allowing all edge virtual channel resources to be devoted to fully adaptive routing that is oblivious to deadlocks. Deadlocks, upon forming, are handled efficiently by nominal router resources dedicated to this purpose. Each router is equipped with a dedicated flit buffer (a Deadlock Buffer), to be used only in the rare case of deadlocks. These Deadlock Buffers are input buffers, central to the router and can be accessed from all neighboring nodes. These essentially form a dedicated deadlock-free lane which can be visualized as a "floating" virtual channel. On deadlock, one of the packets in the cycle is switched to the deadlock-free lane and routed using this resource until it reaches its destination where it will be consumed (sunk) to break the dependency cycle.

Router switch (crossbar) connections are allocated to messages either on a flit-by-flit basis or a packet-by-packet basis. Flit-by-flit allocation requires the crossbar to be reconfigured on every flit transmission (i.e., physical crossbar). Packet-by-packet allocation results in the crossbar being reconfigured anew for every packet transmission as opposed to every flit (i.e., virtual crossbar). *Disha* is applicable to both crossbar allocation policies.

The operation of *Disha* for flit-by-flit crossbar allocation is explained below. For simplicity, we illustrate operation with only one virtual channel per physical channel. Operation is similar with multiple virtual channels. Figure 2a shows a possible initial state of routers R_a, R_b and R_c. Here, packet P_1 is blocked by P_2 and is unable to proceed. Router R_a is unable to send out the *header* of P_1 for T_{out} clock cycles and assumes packet P_1 to be deadlocked. Router R_a now sends out the *header* of P_1 with the corresponding *Status* line asserted. Router R_b places the incoming *header* of P_1 on the Deadlock Buffer, bypassing the normal input buffer currently occupied by packet P_2. Packet P_1 now follows the Deadlock Buffer path through subsequent routers to reach its destination. These actions are indicated in Figure 2b.

The operation for packet-by-packet router switch allocation is explained below. In the absence of deadlocks, the router crossbar is configured as specified by the decision and arbitration logic. However, in a deadlock situation, it could so happen that a deadlocked packet on the Deadlock Buffer requires the same output connection as is currently being used by some other packet from an edge input buffer. Because the crossbar is not reconnected after every flit transmission, the crossbar must be reconfigured to connect the Deadlock Buffer to the output terminal. The decision logic thus needs to remember the state of the crossbar before it was reconfigured using a reconfiguration buffer. Data routed on the Deadlock Buffer is guaranteed to reach its destination. The suspended input buffer is reconnected back to the preempted output buffer once deadlock has cleared using the information stored in the reconfiguration buffer.

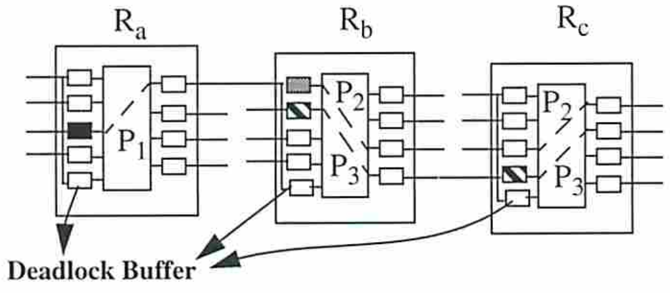


Fig. 2a shows a possible deadlock scenario where P_1 would otherwise wait on P_2 indefinitely.

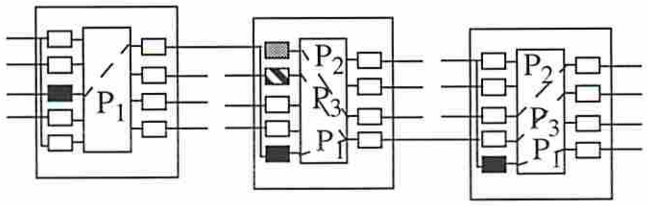


Fig. 2b shows how deadlocks are resolved. P_1 is routed around P_2 on the Deadlock Buffer to break the cycle.

3.0 Definitions and Generalized Theorem for Deadlock Freedom:

A sufficient condition for adaptive routing in wormhole networks to avoid deadlock is proposed in [10]. This theory is extended with a necessary and sufficient condition in [11]. Strictly speaking, those results cannot be directly applied to schemes, such as *Disha,* which use both edge buffers (traditionally referred to as virtual channels) and central buffers. Additionally, the routing function for edge buffers is usually different from that for central buffers. Thus, the routing function must consider the type of buffer where the packet is stored at the current node.

This paper develops a generalized version of the theory in [10] based on the notion of buffers as opposed to channels. This now relaxes router assumptions and provides the framework for designing schemes with both edge and central buffers like *Disha*. The generalized theory is applicable to deadlock avoidance as well as deadlock recovery schemes. It is possible to extend the theory proposed in [11] in the same way, but for the applications considered in this paper, we do not need the additional flexibility afforded by that theory. As we only introduce changes in notation, our extensions do not change the way the theorem is proved. Thus, we will not repeat the proof of the theorem here as it is similar to the one in [10]. Our generalized theorem is presented below, following some definitions.

Definition 1:

An *interconnection network* I is a strongly connected directed multigraph, $I = G(N, C)$. The vertices of this multigraph N represent the set of processing nodes and the arcs C represent the set of communication channels connecting the nodes. Each channel c_i has an associated queue or edge buffer. Additionally, each node n_j contains one or more central buffers q_{j1}, q_{j2},... accessible by all buffers at neighboring nodes. Let Q be the set of central buffers (Deadlock Buffers). Let b_k be an element of the set of buffer resources $B = C \cup Q$.

Definition 2:

An *adaptive routing function* $R : B \times N \rightarrow \Pi(B)$, where $\Pi(B)$ is the power set of B, supplies a set of alternative buffers in neighboring nodes to send a message from the current buffer b_c to the destination node n_d, $R(b_c, n_d) = \{b_1, b_2, ..., b_p\}$. A *selection function* selects a free buffer (if any) from the set supplied by the routing function.

Definition 3:

The routing function R for a given interconnection network I is *connected* iff, for any current buffer and destination node, it is possible to establish a path between them using the buffers belonging to the set supplied by R.

Definition 4:

A *routing subfunction* R_1 for a given routing function R and buffer subset $B_1 \subseteq B$, is a restriction on R, so that it only supplies buffers belonging to B_1. A connected routing subfunction R_1 is able to establish a path between any buffer and any node, including the buffers not supplied by R_1. However, R_1 will only supply buffers from B_1 while establishing the path. More precisely, $\mathrm{R_1(b, n) = R(b, n) \cap B_1, \forall\, b \in B, \forall\, n \in N}$.

Definition 5:

Given an interconnection network I, a routing function R and a pair of buffers $b_i, b_j \in B$, there is a *direct dependency* from b_i to b_j iff b_j can be used immediately after b_i by messages destined for some node n.

Definition 6:

Given an interconnection network I, a routing function R, a buffer subset $B_1 \subseteq B$ which defines a routing subfunction R_1 and a pair of buffers $b_i, b_j \in B_1$, there is an *indirect dependency* from b_i to b_j iff it is possible to establish a path from b_i to b_j for messages destined for some node n. The buffers b_i and b_j are the first and last buffers in that path and the only ones belonging to B_1. As b_i and b_j are not adjacent, some other buffers not supplied by R_1 are used between them.

Definition 7:

A *resource dependency graph* D for a given interconnection network I and a routing function R is a directed graph $D = G(B, E)$. The vertices of D are the buffers of B and the arcs of D are the pairs of buffers (b_i, b_j) such that there is a direct dependency from b_i to b_j.

Definition 8:

The *extended resource dependency graph* D_E for a given interconnection network I and a routing subfunction R_1 of a routing function R, is a directed graph $D_E = G(B_1, E_E)$. The vertices of D_E are the buffers that define the routing subfunction R_1. The edges of D_E are the pairs of buffers (b_i, b_j) such that there is either a direct or an indirect dependency from b_i to b_j.

Theorem 1:

A connected and adaptive routing function R for an interconnection network I is deadlock-free if there exists a subset of buffers $B_1 \subseteq B$ that defines a routing subfunction R_1 which is connected and has no cycles in its extended resource dependency graph D_E.

Proof:

The proof for this theorem is along the same lines as the theorem derived in [10].

4.0 Application of Generalized Theorem to *Disha* Concurrent:

The simple methodology we use for designing *Disha* free from the requirement of mutually exclusive access to the deadlock-free lane is presented here. The main idea is to structure routing on the central Deadlock Buffers such that cyclic dependencies do not occur. Sections 4.1 and 4.2 discuss concurrent recovery as applied to n-dimensional meshes and Sections 4.3 and 4.4 deal with concurrent recovery in n-dimensional toroids.

4.1 Concurrent recovery methodology for 2-cube meshes:

The methodology for concurrent recovery in a 2D mesh is given below; an extension to n-dimensional meshes is presented in the next subsection.

1) Construct a Hamiltonian path on the 2D mesh using Deadlock Buffers similar to the one shown in Figure 3.

2) Number the nodes in sequence along the Hamiltonian path, again as shown in Figure 3.

3) The Deadlock Buffer at a node can be used only by a deadlocked packet at a neighboring node for which the packet's destination has a higher label than the node to which the buffer belongs.

4) A packet, once placed on a Deadlock Buffer, is restricted to using only Deadlock Buffers at subsequent nodes until it is delivered.

5) Successive Deadlock Buffers can be used only in increasing label order.

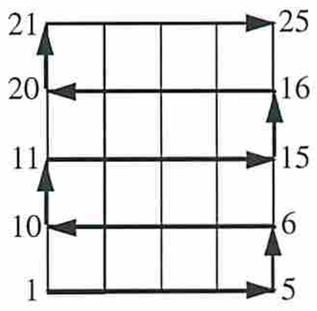


Figure 3: Hamiltonian path for a 5 ×5 mesh.

The Hamiltonian path constructed as described above is equivalent to the high-channel network in [15]. This structured path allows any node to reach any other node whose label is higher than its own. It is easy to see that the Hamiltonian path is acyclic and routing on the Deadlock Buffers is deadlock-free. However, the generalized theory developed in Section 3 cannot be directly applied in proving that routing is deadlock-free as packets routed on the Deadlock Buffers are able to reach higher labelled destinations only, resulting in a non-connected routing subfunction defined on the Deadlock Buffers. With subtle changes made to the routing subfunction, we can form a connected routing subfunction that remains deadlock-free. With this augmented routing subfunction, deadlocked packets can be routed to the destination or some node less than the destination, following which they can use the Deadlock Buffers to be routed to the destination and sunk there. A formal proof is presented below.

Definition 9:

The *label* of a node *i* with coordinates (i_x, i_y), $\forall\ x \in [1, m]$ and $\forall\ y \in [1, n]$, is given by the following Hamiltonian assignment:

$$\pi[i] = \pi[x, y] = \begin{cases} n(x - 1) + y & \text{if x is odd} \\ nx - y + 1 & \text{if x is even.} \end{cases}$$

Row numbers increase from bottom to top and column numbers increase from left to right. This labelling for a 5×5 mesh is illustrated in Figure 3.

To apply our generalized theorem, we make the following assumptions about the implementation of *Disha* Concurrent.

Assumption 1:

Deadlock Buffers are connected in a Hamiltonian path like the one shown in Figure 3. A Deadlock Buffer q_j has the same label as the node *j* it belongs to, i.e., $\pi[q_j] = \pi[j]$.

Assumption 2:

Deadlock Buffers can be used only on a deadlocked situation. The Deadlock Buffer q_j at node *j* can be used by a packet at any neighboring node for which the destination node has a higher node label than the node to which the Deadlock Buffer belongs, i.e., $\pi[q_j] < \pi[d]$.

Assumption 3:

A packet that has been placed on the Deadlock Buffer of a node can use only Deadlock Buffers at subsequent nodes until it is delivered at its destination.

Assumption 4:

Successive Deadlock Buffers can only be used in increasing label orders and routing on the Deadlock Buffers is specified by the routing subfunction $R_{db} : Q \times N \rightarrow Q$ such that $R_{db}(q_j, d) = q_k$, where $\pi[q_k] = \max\{\pi[q_i] : \pi[q_i] \leq \pi[d]\}$ and q_i is a Deadlock Buffer at a neighboring node *i*.

Assumption 5:

The routing function R which supplies the set of channels C is fully adaptive and minimal.

Constructing a Connected Routing Subfunction:

Let Q be the set of Deadlock Buffers. Only those destinations whose node labels are greater than the current node label can be reached by packets through Deadlock Buffers. Therefore, Deadlock Buffers alone do not form a *connected* routing subfunction. To these, we add an edge buffer at every node. Each edge buffer has an associated unidirectional channel and, consequently, can be specified in terms of these channels. Hence, we add a unidirectional channel from every node *c* to its neighbor *n* such that $\pi[n] = \min\{\pi[i]\}$, where i is a neighboring node of c and $\pi[c] \neq 1$. These channels C_{edge} are shown in Figure 4 below. Let $B_1 = Q \cup C_{edge}$, which are the buffers supplied by the routing subfunction R_1.

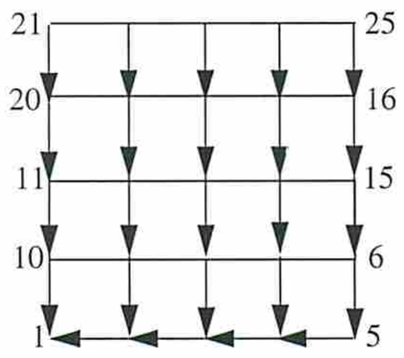


Figure 4: Channels added to make R_1 connected.

It should be noted that C_{edge} are existing edge buffers ($C_{edge} \subset C$) and not additional channels. They are included here only to form a connected routing subfunction to prove that *Disha* is deadlock-free. As Assumption 5 states, there is no restriction on routing using C_{edge}, other than that routing should be minimal (limited misrouting is also possible, but we do not examine this case).

Lemma 1:

The routing subfunction R_1 is connected.

Proof:

$R_1 : B \times N \rightarrow B_1$ and $B_1 = Q \cup C_{edge}$. Also, $R_1(b, n) = R(b, n) \cap B_1$. Given any pair of nodes (c, d) it is possible to establish a path from *c* to *d* using buffers from the set Q supplied by R_{db} if $\pi[d] > \pi[c]$. If on the other hand $\pi[d] < \pi[c]$, then it is possible to establish a path from c to some node n_1 such that $\pi[n_1] \leq \pi[d]$ using edge buffers $C_{edge.}$ If $n_1 \neq d$, a path from n_1 to *d* can then be established using central buffers supplied by R_{db}. Thus, R_1 is connected.

Lemma 2:

Any cycle in the extended resource dependency graph of R_1 does not involve buffers belonging to the set Q.

Proof:

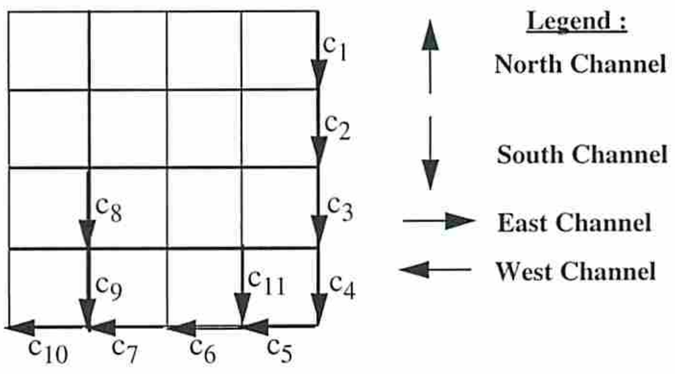
It follows from Assumption 2 that if a buffer q_i is supplied by R_{db} then all subsequent buffers must be supplied by R_{db}. Therefore, we cannot find a pair of buffers (q_i, q_j) supplied by R_{db} such that the buffers between them are not supplied by R_{db}. Consequently there are no indirect dependencies involving buffers belonging to Q. Additionally, taking into account Assumption 3, there are no direct dependencies from a buffer belonging to Q to other buffers that do not belong to Q. Also, from Assumption 4, packets are routed on the Deadlock Buffers in increasing label order. Thus, R_{db} by itself is deadlock-free. Hence, any cycle in the extended resource dependency graph of R_1 cannot involve buffers belonging to Q.

Lemma 3:

The routing subfunction R_1 has no cycles in its extended resource dependency graph.

Proof:

Buffers Q cannot be involved in any cycle (Lemma 2). Let us therefore only analyze the buffers C_{edge}. These buffers are involved in both direct and indirect dependencies. There are both direct and indirect dependencies from South channels to South channels in lower rows along the same column, South channels to West channels in the first row, and West channels to West channels in the first row. (The indirect dependencies arise when not all edge buffers in the South channels and West channels along first row belong to C_{edge}.) There are also indirect dependencies between South channels to South channels in lower rows along different columns. Examples of these are shown below. There is an indirect dependency between c_1 and c_3, c_3 and c_5, c_5 and c_7, c_3 and c_{11}. There are direct dependencies between c_8 and c_9, c_9 and c_{10}, c_7 and c_{10}. There are no other dependencies as long as packets follow only minimal paths. Examples of these dependencies are shown in Figure 5.



Channels $c_1, c_3, c_5, c_7, c_8, c_9, c_{10}, c_{11} \in C_{edge}$

Channels $c_2, c_4, c_6 \in (C - B_1)$.

Figure 5: Direct and Indirect Dependencies.

There are no indirect dependencies from South channels to South channels in higher rows, or from West channels to South channels. Because $B_1 = Q \cup C_{edge}$, it follows that there are no cycles in the extended resource dependency graph. Figure 6 shows the extended resource dependency graph for C_{edge} of a 3×4 mesh; it is acyclic.

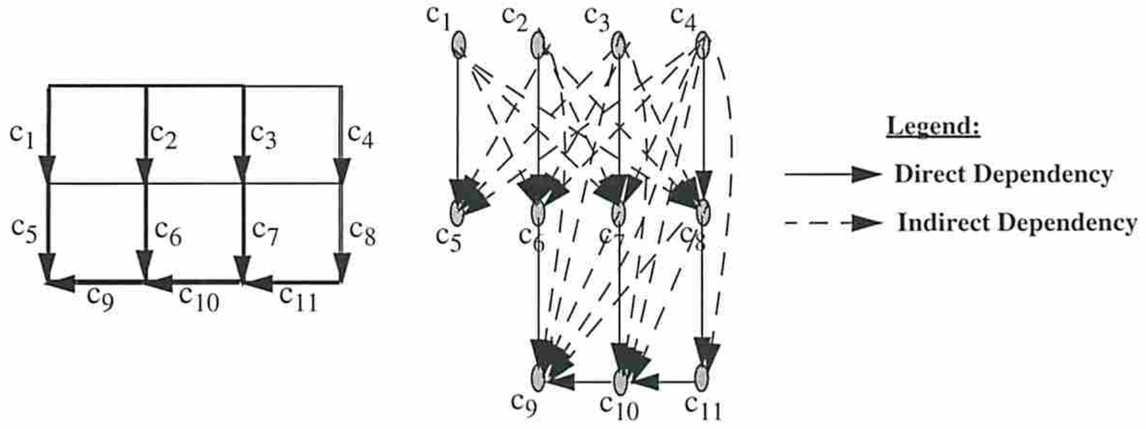


Figure 6: The extended resource dependency graph for a 3×4 mesh.

Theorem 2:

"Under minimal routing, a two dimensional mesh network can safely recover from deadlocks when deadlocked packets are routed *concurrently* on the Deadlock Buffers".

Proof:

From Lemma 1, the routing subfunction R_1 is connected. Lemma 3 shows that there are no cycles in the extended resource dependency graph. Therefore, from Theorem 1, it follows that *Disha* is deadlock-free.

■

Deadlocks can be detected with a time-out. If a packet is unable to make progress for a time duration corresponding to the time-out, it presumes that it is deadlocked. *Disha* recovers from deadlocks regardless of its value, provided that it is finite. A time-out of zero would correspond to deadlock avoidance (as opposed to recovery). However, insufficient time-outs are not efficient as simulation results in Section 5.1 will demonstrate.

4.2 Concurrent Recovery Methodology for k-ary n-cube meshes:

It is possible to extend this scheme to k-ary n-cube meshes. We consider the example of a 3D mesh; extending to any k-ary n-cube mesh is similar. This 3D mesh can be visualized as a number of x-y planes stacked one above the other. One possible Hamiltonian path starts in the first x-y plane and eclipses all nodes in that plane before moving to the next one below as shown in Figure 7a. The path in a plane is similar to the one shown in Figure 3, though the flow is in opposing directions in alternate layers. The routing subfunction makes use of all the Deadlock Buffers, all Z^+ channels, Y^- channels in the first (topmost) plane and X^- channels in the first row of the topmost plane as shown in Figure 7b. All nodes in lower planes can be reached through the Deadlock Buffers (Q). To reach nodes in higher planes, the Z^+ channels of C_{edge} (Y^- and X^- if necessary too) could be used to reach the destination or some intermediate node with a lower label than the destination label. A path from this intermediate node to the destination can be established through Q. The routing subfunction defined on $B_1 = Q \cup C_{edge}$ is therefore connected, and Theorem 1 developed previously can similarly be applied to verify that *Disha* Concurrent for any n-dimensional mesh network is also deadlock-free.

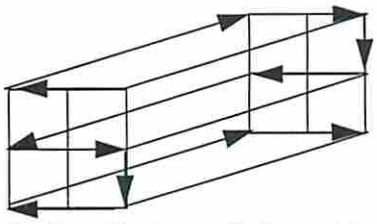


Figure 7a: Hamiltonian path formed by DBs.

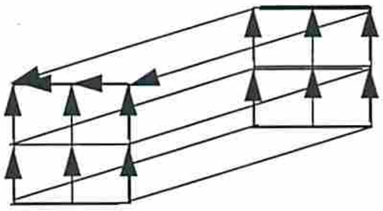


Figure 7b: Channels added to make R_1 connected.

Theorem 3:

"Under minimal routing, any n-dimensional mesh network can safely recover from deadlocks when deadlocked packets are routed *concurrently* on the Deadlock Buffers".

4.3 Concurrent recovery methodology for k-ary n-cube toroids:

It is easy to see that the above methodology does not work for tori. Wrap-arounds introduce dependencies between South channels in lower rows and those higher up. Similarly dependencies are introduced between West channels at the far left and those at the far right. The extended resource dependency graph is no longer acyclic. For toroids, we therefore use two Deadlock Buffers and adopt the methodology given below. We formally present a proof for the case of a two dimensional toroid; extensions to n-dimensional toroids follow along similar lines.

1) Divide the Deadlock Buffers into two groups - one to be used if current node label is greater than destination node label and the other if current node label is less than destination node label.

2) With each Deadlock Buffer class, form separate Hamiltonian paths in opposing directions to one another as shown in Figure 8. These Hamiltonian paths are equivalent to the high and low channel networks in [15].

3) Number the nodes on each Hamiltonian path in sequence.

4) Packets using Deadlock Buffers are restricted to belong to either the high or the low channel network alone and follow Deadlock Buffers strictly in increasing or decreasing label order respectively until delivered.

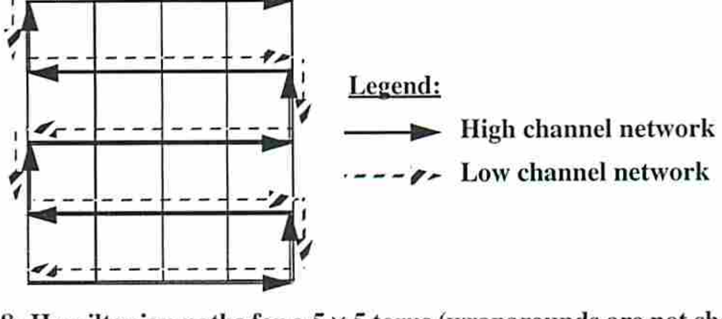


Fig. 8: Hamiltonian paths for a 5 × 5 torus (wraparounds are not shown).

The proposed implementation of *Disha* Concurrent with two Deadlock Buffers is deadlock-free. Packets deadlocked on the edge buffers use either the high or low channel network formed by Deadlock Buffers in strictly increasing or decreasing label order. There are no cycles. A formal proof is given below.

Assumption 6:

Deadlock Buffers are connected in two distinct Hamiltonian paths forming high and low channel networks as shown in Figure 8. Let Q_h and Q_l represent the set of Deadlock Buffers constituting the high and low channel networks respectively. Deadlock Buffer at node *j* belonging to the high channel network is denoted by q_{jh} and that belonging to the low channel network is denoted by q_{jl}. Let $Q = (Q_h \cup Q_l)$ be the set of Deadlock Buffers.

Assumption 7:

A Deadlock Buffer q_{jx} has the same label as the node *j* it belongs to, i.e., $\pi[q_{jx}] = \pi[j]$.

Assumption 8:

Deadlock Buffers can be used only on deadlock. The Deadlock Buffer q_{jh} at node *j* can be used by a packet at any neighboring node for which the destination node has a higher node label than the node to which the Deadlock Buffer belongs, i.e., $\pi[q_{jh}] < \pi[d]$. Likewise, the Deadlock Buffer q_{jl} at node *j* can be used by a packet at any neighboring node for which the destination node has a lower node label than the node to which the Deadlock Buffer belongs, i.e., $\pi[q_{jl}] > \pi[d]$.

Assumption 9:

A packet that has been placed on the Deadlock Buffer of a node can use only Deadlock Buffers at subsequent nodes until it is delivered at its destination.

Assumption 10:

Successive Deadlock Buffers can only be used in increasing or decreasing label orders and routing on the Deadlock Buffers is specified by the routing subfunction $R_{dbx} : Q_x \times N \rightarrow Q_x$. If $\pi[q_{jh}] < \pi[d]$, then $R_{dbh}(q_{jh}, d) = q_{kh}$, where $\pi[q_{kh}] = \max\{\pi[q_{ih}] : \pi[q_{ih}] \leq \pi[d]\}$ and q_{ih} is a Deadlock Buffer at a neighboring node *i*. Similarly, if $\pi[q_{jl}] > \pi[d]$, then $R_{dbl}(q_{jl}, d) = q_{kl}$, where $\pi[q_{kl}] = \min\{\pi[q_{il}] : \pi[q_{il}] \geq \pi[d]\}$ and q_{il} is a Deadlock Buffer at a neighboring node *i*. The routing function thus restricts packets to belong to either the high or low channel networks alone. In other words, $R_{db}(q, n) = R_{dbh}(q, n) \cup R_{dbl}(q, n)$.

Assumption 11:

The routing function R which supplies the set of channels C is fully adaptive. Routing on edge buffers can even be non-minimal provided that no node is visited twice by the same packet.

Definition 10:

Let $B_1 = Q = (Q_h \cup Q_l)$ be the set of buffers supplied by the routing subfunction R_1.

Lemma 4:

The routing subfunction R_1 is connected.

Proof:

$R_1 : B \times N \rightarrow B_1$ and $B_1 = (Q_h \cup Q_l)$. Also, $R_1(q, n) = R_{db}(q, n) = R_{dbh}(q, n) \cup R_{dbl}(q, n)$. Given any pair of nodes (c, d) it is possible to establish a path from *c* to *d* using buffers from the set Q_h, supplied by R_{db}, if $\pi[d] > \pi[c]$ and buffers from the set Q_l, again supplied by R_{db}, if $\pi[d] < \pi[c]$. R_1 is thus connected.

Lemma 5:

Any cycle in the extended resource dependency graph of R_1 does not involve buffers belonging to the set Q.

Proof:

It follows from Assumption 8 that if a buffer q_{jx} is supplied by R_{dbx} then all subsequent buffers must be supplied by R_{dbx}. Therefore, we cannot find a pair of buffers supplied by R_{dbx} such that the buffers between them are not supplied by R_{dbx}. Consequently, there are no indirect dependencies involving buffers belonging to Q. Additionally, taking into account Assumption 9, there are no direct dependencies from a buffer belonging to Q_x to other buffers that do not belong to Q_x. Also, from Assumption 10, packets are routed on the Deadlock Buffers in strictly increasing or decreasing label orders. Thus, R_{dbx} by itself is deadlock-free. Hence, any cycle in the extended resource dependency graph of R_1 cannot involve buffers belonging to Q.

Lemma 6:

The routing subfunction R_1 has no cycles in its extended resource dependency graph.

Proof:

Buffers Q cannot be involved in any cycle (Lemma 2). Because $B_1 = Q$, it follows that there are no cycles in the extended resource dependency graph.

Theorem 4:

"Two dimensional toroidal networks can safely recover from deadlocks when deadlocked packets are routed *concurrently* on the Deadlock Buffers".

Proof:

From Lemma 1, the routing subfunction R_1 is connected. Lemma 3 shows that there are no cycles in the extended resource dependency graph. Therefore, from Theorem 1, it follows that *Disha* is deadlock-free.

■

Theorem 5:

"N-dimensional toroidal networks can safely recover from deadlocks when deadlocked packets are routed *concurrently* on the Deadlock Buffers".

4.4 Increasing routing flexibility in toroids:

The methodology adopted thus far restricts deadlocked packets to use either the low or high-dimension networks. Dependencies between the two networks are not allowed. Flexibility in routing on the Deadlock Buffers can be enhanced and routing optimized by adopting a different labelling strategy. The base Hamiltonian paths are still chosen to be exactly opposing one another. Numbering of buffers (as opposed to nodes) is in one continuous stretch. For the 5 by 5 torus in Figure 9 (wraparounds not shown), buffer labelling would be from 1 to 50 (with corner node labels shown). Packets always use buffers in increasing label order, and it is easy to see that routing under this labelling is also deadlock-free

.

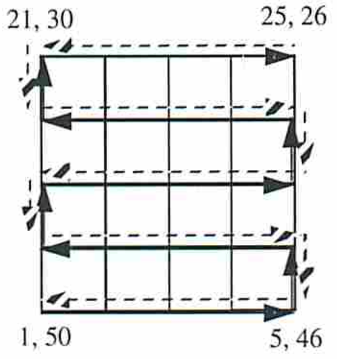


Fig. 9: Alternative node Labelling

This, however, can increase flexibility as shown in Figure 10. With the previous labelling, if the channel marked "X" is faulty, then destination 'D_2'would be unreachable from 'S'. With the new labelling, a packet could use Deadlock Buffers 7, 37, 38, 39 and 40 to reach 'D_2'. Similarly, there are now alternate Deadlock Buffer paths to 'D_1' from 'S' - (7, 14), (7, 37) and (15, 37). This added flexibility increases the fault-tolerance capabilities of *Disha* Concurrent.

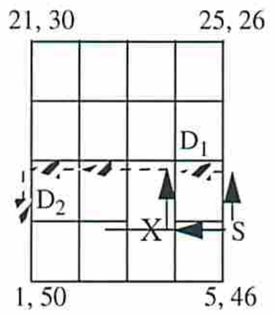


Fig. 10: Increasing routing flexibility on Deadlock Buffers

In summary, it is possible to have concurrent recovery with *Disha*. This eliminates the Token required for implementing mutual exclusion in sequential recovery. It also allows recovery from cycles simultaneously. K-ary n-cube meshes require just a single Deadlock Buffer per router whereas toroidal k-ary n-cubes require two Deadlock Buffers. Routing on meshes is restricted to be minimal while that on toroids can be non-minimal as well. However, in the case of toroids, deadlocked packets suffer from increased hop distances as the recovery path is non-minimal. The cost of implementing two Deadlock Buffers is also slightly more. However, we do not expect the router delay to be affected significantly.

5.0 Performance Results:

Simulations are run on a 16 by 16 (256 node) network with four virtual channels per physical channel. Messages are 32 flits long. A buffer depth of two is selected (shallow buffers keep the router simple and reduce clock cycle time). Equal clock cycle is assumed for all schemes being compared. Performance graphs show normalized throughput versus average latency. (Normalized throughput of 1.0 indicates that 50% of uniform traffic crosses the bisection of the network).

5.1 Comparisons for the Mesh:

Dimension ordered routing (*DOR*) is non-adaptive. *Duato* is based on the conditions derived in [10]. *Disha-Seq* traces sequential recovery in *Disha* [1]. Concurrent recovery (*Disha-Con*) based on results in Section 4.2 is simulated for two different time-outs of 8 and 1000.

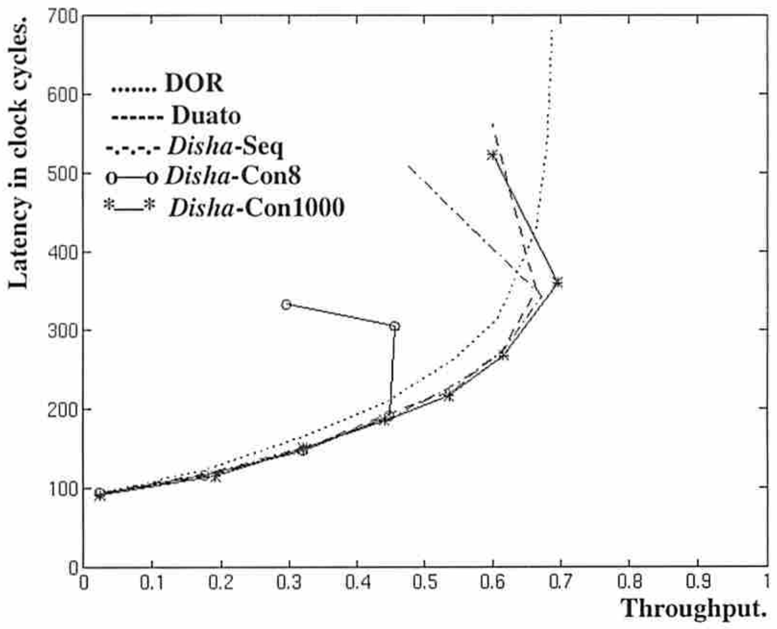


Fig. 11: Comparison for Uniform Traffic.

Under uniform traffic, each node sends packets to all other nodes with equal probability. Performance is very similar for all schemes except *Disha-Con8,* as can be seen from Figure 11. The insufficient time-out in *Disha-Con8* causes false deadlocked (congested) packets to saturate the DBs, causing the DBs to be a bottleneck. As deadlocked packets are restricted to remain on DBs, performance suffers. The fact that the non-adaptive *DOR* performs almost as well as the fully adaptive *Disha* and *Duato* is not surprising as it preserves the traffic's uniformity. A peak throughput of 0.7 is obtained by *Disha-Con1000.*

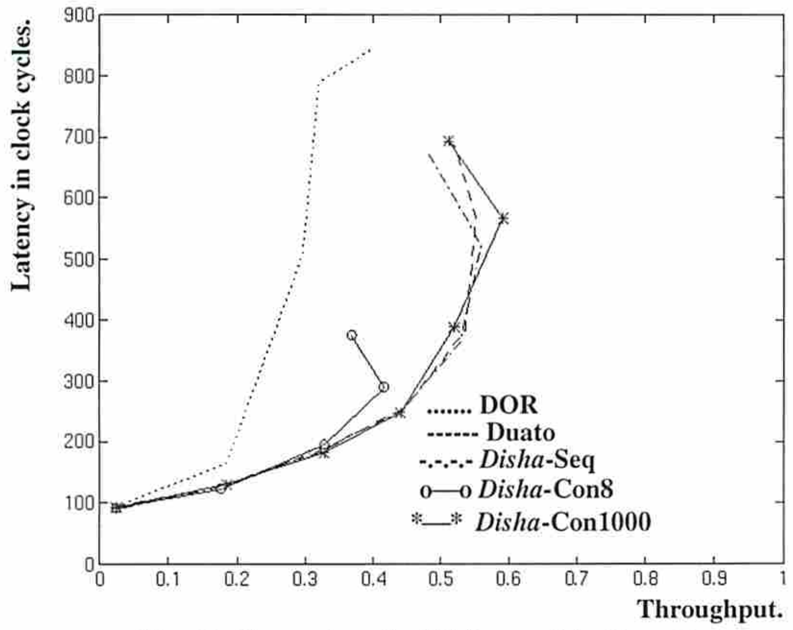


Fig. 12: Comparison for Bit-Reversal traffic pattern.

We now compare the performance of our new algorithm for non-uniform traffic patterns. Performance results for the bit-reversal traffic pattern are shown in Figure 12. Here, a node with coordinates $a_{n-1}, a_{n-2}, ..., a_1, a_0$ sends messages to the node with coordinates $a_0, a_1, ..., a_{n-2}, a_{n-1}$. With this traffic, nodes along a given row send messages to nodes along a given column. We now see the deterioration in performance of the non-adaptive *DOR* with respect to the other fully adaptive schemes. *Duato*, *Disha-Seq* and *Disha-Con1000* are clustered in performance while *Disha-Con8* saturates early. This trend continues in other non-uniform traffic patterns simulated as well.

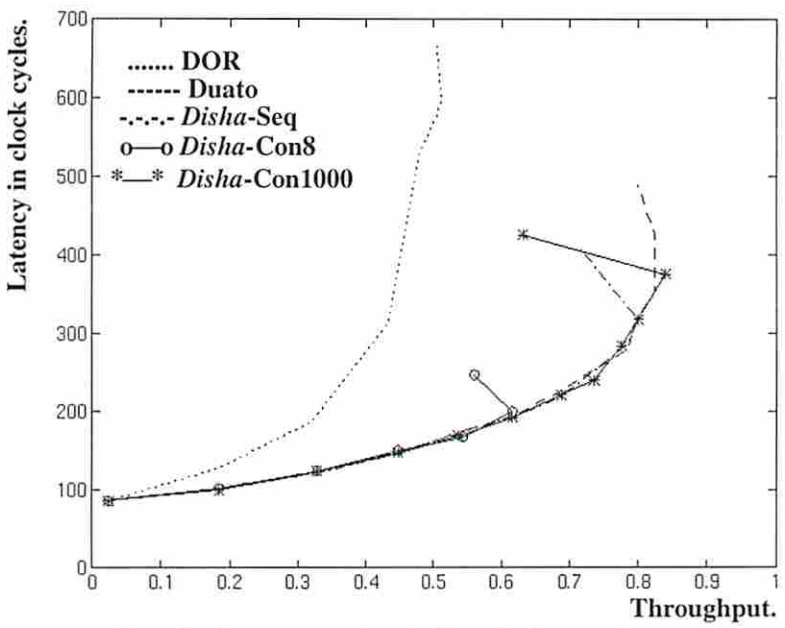


Fig. 13: Comparison for Perfect Shuffle traffic pattern.

For the perfect shuffle traffic pattern, a node with coordinates a_{n-1}, a_{n-2}, ..., a_1, a_0 sends to node a_{n-2}, ..., a_1, a_0, a_{n-1}. *Duato*, *Disha-Seq* and *Disha-Con1000* perform almost identically. Peak throughput is around 0.85 as shown in Figure 13.

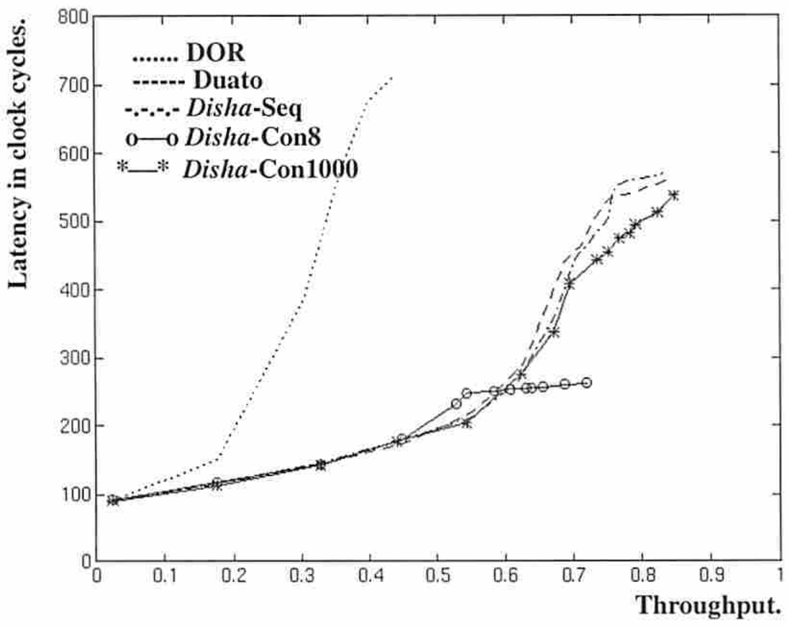


Fig. 14: Comparison for Matrix Transpose traffic pattern.

Finally comparisons are drawn in Figure 14 for matrix-transpose, where a node with coordinates (x, y) has node (y, x) as its destination. Interestingly here, *Disha-Con8* incurs very low latencies although its peak throughput is about 15% less than the other fully adaptive schemes.

5.2 Comparisons for the Torus:

This section evaluates the performance of the concurrent *Disha* scheme for a Torus. The torus type topology necessities two central buffers in *Disha-Con*. Performance in *Disha* is evaluated for a single time-out of 128. Evaluations are done for the same traffic patterns as in Section 5.1. Our results show that *Disha-Con* consistently outperforms all other schemes.

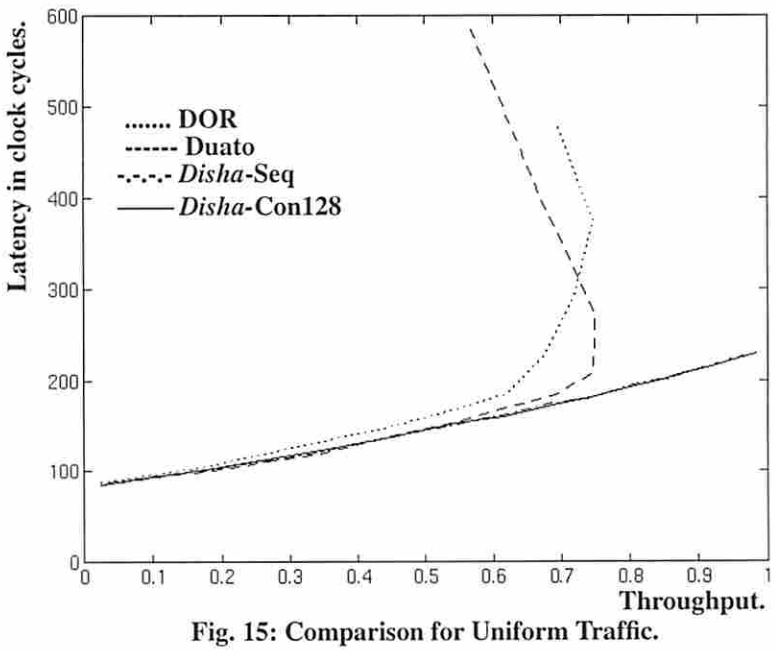


Fig. 15: Comparison for Uniform Traffic.

Results in Figure 15 indicate that peak throughput in *Disha* (Seq and Con) is 25% greater than *DOR* and *Duato* and, unlike the other two, is sustained. Uniform traffic accounts for the enhanced performance observed in *DOR*.

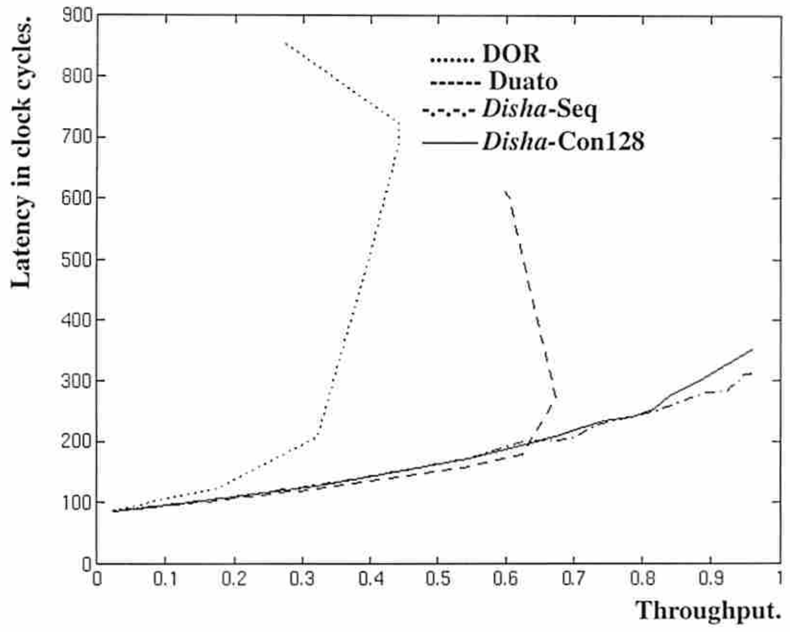


Fig. 16: Comparison for Bit-Reversal traffic pattern.

For bit-reversal, again, both the *Disha* schemes produce sustained throughputs far greater than *Duato* and *DOR*. Performance of *DOR* relative to *Duato* drops off with non-uniformity in traffic. Figure 16 shows the performance curves.

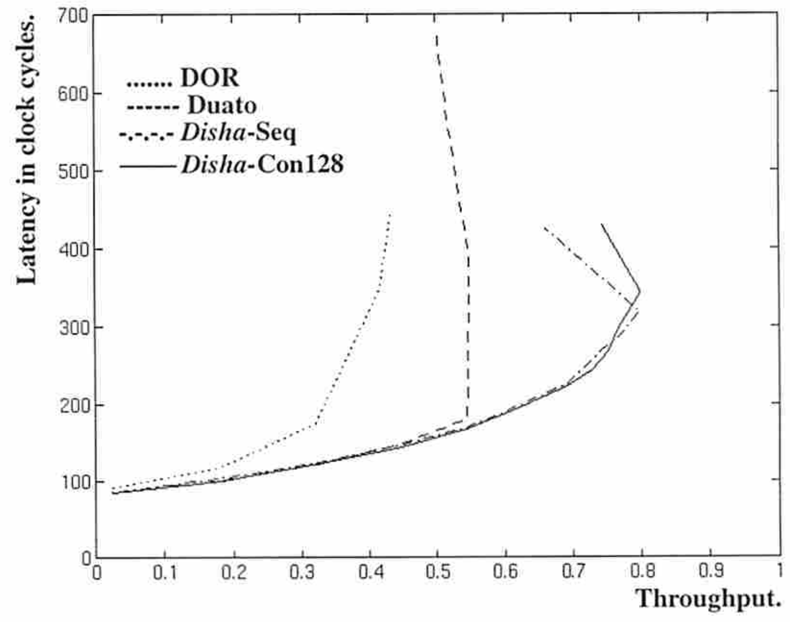


Fig. 16: Comparison for Perfect Shuffle traffic pattern.

Results for perfect-shuffle are plotted in Figure 17. *Disha* provides a 45% improvement in peak throughput compared to *Duato* and about 200% increase over the non-adaptive *DOR*. The peak throughput is 0.8 but is not sustained.

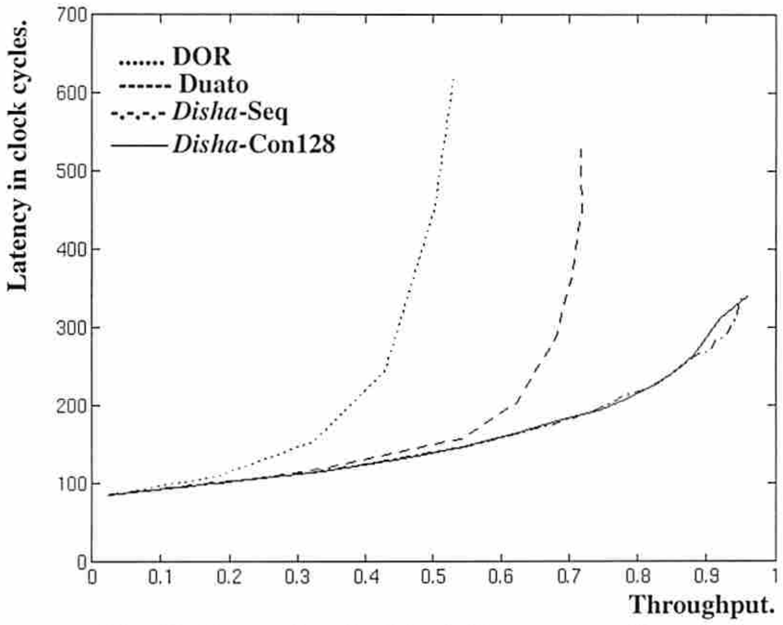


Fig. 18: Comparison for Matrix Transpose traffic pattern.

Finally, comparisons are drawn for matrix-transpose in Figure 18. Sustained peak throughputs of about 30% greater than the fully adaptive *Duato* again demonstrates the superior performance obtained by our new approach and justifies the design methodology.

5.3 Discussion:

Simulations indicate that, for concurrent recovery with *Disha*, superior performance hinges more significantly on accurate deadlock detection. All packets presumed deadlocked are eligible to recover and are immediately routed on the deadlock buffer lane (if free at next node). Because packets must remain on this lane until reaching their destinations, the deadlock buffer lane quickly becomes congested with non-deadlocked packets if many falsely-detected deadlocks are acted upon. This is seen in the *Disha-Con8* performance results. Here, the time-out interval is insufficient to filter out blocked packets due to network congestion (false deadlocks) from blocked packets due to cyclic dependencies (true deadlocks). The larger time-out (*Disha-Con1000*) better filters out false deadlock detections, selectively allowing only those packets truly in need of deadlock recovery to use the recovery lane. Simulation runs for the mesh show as much as 50% improvement in performance for *Disha* Concurrent with more accurate deadlock detection. Similar results are seen for the torus. This suggests that more accurate deadlock detection mechanisms beyond that of time-outs may be warranted for concurrent recovery.

Sequential recovery [2] is not as sensitive to the detection of true deadlocks. Packets presumed deadlocked are eligible to recover via the deadlock buffer lane only after acquiring the Token (for mutual exclusive access). The deadlock buffer lane can never become congested as only one packet is allowed to route on it at a time. Blocked packets due to congestion are likely to be routed by the time the Token arrives, increasing the probability that truly deadlocked packets are recovered. The selectivity implemented by the Token is actually seen to be beneficial in this case.

If a non-deadlocked packet acquires the Token, its removal from the congested area of the network via the deadlock-free lane only further improves performance as deadlocks are rare anyway. Hence, for sequential recovery with *Disha*, a proper choice of time-out is one which minimizes network congestion to improve performance, not one which minimizes access to the deadlock-free lane.

Sequential recovery has similar throughput-latency curves as concurrent recovery and in some cases cannot even be distinguished. As the two are based on the same progressive recovery approach, the slight differences between them are attributed to the concurrency (or lack of it) allowed in recovering from deadlocks. Because deadlocks are rare, little difference is seen between the two schemes. Therefore, the performance of true-fully-adaptive *Disha (Seq and Con) recovery* is superior to the fully adaptive *Duato* and the non-adaptive *DOR* deadlock avoidance schemes. It should be noted that our new approach has completely eliminated the Token resource which could pose as a central point of failure. Hence, the improved performance comes with a more feasible, less costly implementation.

For the case of the mesh, *Duato* performs almost as well as *Disha* while for a torus, a marked improvement is seen for *Disha*. Although, both schemes are fully adaptive, the choice of available virtual channels to packets at nodes explains the performance difference. With four virtual channels, when source and destination are not along the same row or column, non-deadlocked packets have a choice of 8 virtual channels in *Disha* for both mesh and torus. In *Duato*, however, packets have a choice of 7 virtual channels for a mesh but only 5 for the torus. This explains why *Duato* performs almost as well as *Disha* in a mesh type network but fares poorly when the network is a torus. Notice that, in *Duato*, as the number of dimensions is increased, the available choice of virtual channels relative to the number of virtual channels present decreases. The performance of *Duato* in comparison to *Disha* for both mesh and tori would drop. The same effect would be observed when fewer than four virtual channels per physical channel are used. The performance of *DOR* relative to *Disha* would drop even more as dimensions are increased. Table 1 compares various schemes using Degree of Adaptivity (DoA) as a metric. The calculations are shown for a 256 node (16 by 16) torus with four virtual channels/physical channel. On the average, there are 4,587,520 virtual channel paths from source to destination, where DoA is defined as

$$\text{Degree of Adaptivity} = \frac{\text{allowed number of minimal virtual channels paths}}{\text{possible number of minimal virtual channels paths}}.$$

Table 1: Disha results in 100% adaptivity for no deadlocks (0% for deadlocked packets).

Routing Scheme	Adaptivity	Min. VCs	Choice / 8	DoA
Deterministic (DOR) [7]	None	2	2	$\cong 0\%$
Turn Model [18]	Partial	2	3 (Ave.)	0.2% (Ave.)
Planar Adaptive Routing [5]	Partial	6 (constant)	2	$\cong 0\%$
Linder & Harden [17]	Full	6 (grows exp.)	2	$\cong 0\%$
Dally & Aoki (Dynamic) [9]	Full	3	(4-A; 1-D)	0.4% (Max)
Duato (Sufficient) [10]	Full	3	5	3%
Disha	***True* Full**	**0**	**8**	**100% / 0%**

6.0 Conclusions and Future Work:

The performance of interconnection networks can be improved by pipelining packet transmission and using adaptive routing. Wormhole switching, which is an efficient pipelined switching technique, is unfortunately prone to deadlock because each packet usually holds several channels simultaneously. Until recently, most proposals have focussed on deadlock avoidance.

As deadlocks are relatively rare, deadlock recovery techniques can be used as a viable alternative to deadlock avoidance. The main advantage of recovery techniques is that no resources are "wasted" on avoiding situations that generally occur infrequently at best. Previously proposed recovery techniques are based on regressive abort and retry, thus increasing latency considerably. *Disha* is a unique recovery technique in that it does not kill packets on deadlocks. Instead, it supplies alternative paths to deliver deadlocked packets progressively. Unlike deadlock avoidance techniques, these paths are not dedicated ones. This results in increased sharing of network resources for enhanced efficiency and improved performance as confirmed in simulations [2].

Previous proposals for *Disha* were based on sequential recovery. This could limit the applicability of *Disha* to only those cases where deadlocks are, in fact, infrequent. In this paper, we propose a first extension that allows simultaneous deadlock recovery. N-dimensional meshes require no more buffer resources than previous proposals for sequential recovery. Hence, the underlying scheme is unchanged. N-dimensional toroids as of now require an additional central buffer. Another significant contribution of this paper is the development of a generalized theory for deadlock freedom applicable to both deadlock avoidance as well as deadlock recovery. The theory of deadlock avoidance proposed in [10] is extended to be applicable to routers with a combination of central and edge buffers. This generalized theory is used to prove that *Disha* Concurrent is deadlock-free.

There are many directions that can be explored in extending this work. It might be possible to accomplish concurrent recovery in toroidal n-cubes with a single Deadlock Buffer resource per router. Another interesting area to explore in the future is methods to increase the fault-tolerance capabilities of *Disha* so that, in the presence of faults, there is at least one alternate path on the deadlock-free lane from any node to any other node. This is already true in *Disha* for non-deadlock-free lanes and deadlock-free lanes for which the current node and destination node are not along the same line. Research into more accurate deadlock detection mechanisms is also needed.

References:

[1] Anjan K. V. and Timothy Mark Pinkston. *DISHA*: A Deadlock Recovery Scheme for Fully Adaptive Routing. In *Proceedings of the 9th International Parallel Processing Symposium, IEEE* Computer Society, pages 537-543, April 1995.

[2] Anjan K. V. and Timothy Mark Pinkston. An Efficient, Fully Adaptive Deadlock Recovery Scheme: *DISHA*. In *Proceedings of the 22nd International Symposium on Computer Architecture, IEEE* Computer Society, June 1995.

[3] K. Aoyama. Design Issues in Implementing an Adaptive Router. *Master's Thesis, University of Illinois, Department of Computer Science*, 1304 W. Springfield Avenue, Urbana, Illinois., January 1993.

[4] R. V. Boppana and S. Chalasani. A Comparison of Adaptive Wormhole Routing Algorithms. In *Proceedings of 20th International Symposium on Computer Architecture, IEEE* Computer Society, pages 351-360, 1993.

[5] Andrew A. Chien and J.H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. In *Proceedings of the 19th International Symposium on Computer Architecture, IEEE* Computer Society, pages 268-77, May, 1992.

[6] Andrew A. Chien. A cost and performance model for k-ary n-cube wormhole routers. In *Proceedings of Hot Interconnects Workshop*, August 1993.

[7] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5), pages 547-553, May, 1987.

[8] W. Dally. Virtual channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194-205, March, 1992.

[9] W. Dally and H. Aoki. Deadlock-free Adaptive Routing in Multicomputer Networks using Virtual Channels. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 4, pages 466-475, April, 1993.

[10] J. Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems,* 4(12):1320-1331, December 1993.

[11] J. Duato. A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks. In *Proceedings of the International Conference on Parallel Processing*, pages I142-I149, August, 1994.

[12] J. Duato. Improving the Efficiency of Virtual Channels with Time-Dependent Selection Functions. In *Proceedings of Parallel Architectures and Languages Europe 92,* June 1992.

[13] Patrick T. Gaughan and Sudhakar Yalamanchili. Adaptive Routing Protocols for Hypercube Interconnection Networks. *IEEE Computer*, pages 12-22, May 1993.

[14] J. Kim, Z. Liu and A. Chien. Compressionless Routing: A framework for adaptive and fault-tolerant routing. In *Proceedings of the 21st International Symposium on Computer Architecture, IEEE* Computer Society, pages 289-300, April 1994.

[15] Xiaola Lin and Lionel Ni. Deadlock-Free Multicast Wormhole Routing in Multicomputer Networks, In *Proceedings of the 18th International Symposium on Computer Architecture, IEEE* Computer Society, pages 116-125, May 1990.

[16] Xiaola Lin et al. Deadlock-Free Multicast Wormhole Routing in 2D-Mesh Multicomputers. In *Proceedings of the 5th International Conference on Parallel Processing*, August 1991.

[17] D. Linder and J. Harden. An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary n-cubes. *IEEE Transactions on Computers*, C-40(1): 2-12, January 1991.

[18] L. Ni and C. Glass. The Turn Model for Adaptive Routing. In *Proceedings of the 19th International Symposium on Computer Architecture, IEEE* Computer Society, 20(2):278-287, May, 1992.

[19] Lionel Ni and Philip Mckinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, pages 62-76, February 1993.