Generalized Theory for Deadlock-Free Adaptive Wormhole Routing and its Application to *Disha* Concurrent

Anjan K. V.
Pyramid Technology Corp.
3860 N. First Street
P.O. Box 649013
San Jose, CA 95164-9013
anjan@pyramid.com

Timothy Mark Pinkston
Electrical Engg. - Systems Dept.
University of Southern California
3740 McClintock Ave., EEB-208
Los Angeles, CA 90089-2562
tpink@charity.usc.edu

José Duato
Facultad de Informatica
Univ. Politécnica de Valencia
P.O. Box 22012
46071 - Valencia, SPAIN
jduato@pleiades.upv.es

Abstract:

This paper generalizes a theory for deadlock-free adaptive wormhole routing by considering a mixed set of resources: edge and central buffers. This generalized theory is then applied to a concurrent version of Disha deadlock-recovery which relaxes the sequential recovery requirement for simultaneous recovery from deadlocks. The proposed extension to Disha does not necessitate any additional resource cost; rather, it serves to eliminate the requirement of mutual exclusive access to the deadlock-free lane implemented by a Token. With this extension, Disha Concurrent remains applicable to any topology with a Hamiltonian path including k-ary n-cube networks and is also applicable to tree-based networks.

1.0 Introduction:

Performance of parallel processor systems hinges on effective utilization of system resources. Communication efficiency of the interconnection network can be enhanced by incorporating wormhole switching and adaptive routing. This potent combination is unfortunately susceptible to deadlocks. Traditional schemes based on avoiding deadlock limit the adaptivity of the routing algorithm [4, 12]. 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 [11]. However, additional virtual channels increase the complexity of the router crossbar and the virtual channel controller, leading to longer router clock cycle time and a consequent cost/performance trade-off [5]. Other avoidance schemes [6, 7] designate some virtual channels specifically for the prevention of deadlocks while allowing fully adaptive routing on others. This also leads to reduced utilization of the channels devoted to deadlock avoidance.

Deadlock recovery as a viable alternative to avoidance has only recently begun to gain consideration. Prior research has shown that deadlocks are generally infrequent [2, 9]. 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 [9] 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" redirection of deadlocked packets.

Figure 1 shows a flow diagram of the *Disha* approach. As shown, *Disha* permits unrestricted routing on all existing virtual channels (i.e., edge buffers), and, thus, results *in true* fully adaptive routing. If none of these channels are free during this routing cycle, the packet blocks. After several attempts to route the packet, the router may determine that this packet is potentially deadlocked. At this point, a decision is made as to the eligibility of this packet to progressively recover from deadlock using the recovery path. As only one of the packets involved in a deadlock needs to be eliminated from the dependency cycle to break the deadlock, a packet either uses the recovery path or will eventually be able to use one of the normal edge buffers for routing (i.e., deadlock broken by some other packet).

Progressive recovery from deadlocks is through a flit-sized Deadlock Buffer central to each router which can be accessed from all neighboring nodes. System-wide, these buffers form a deadlock-free lane which can be visualized as a "floating" virtual channel. On the event of a potential deadlock situation, one of the packets in the cycle is switched to the deadlock-free lane and is routed along this path until it reaches its destination, where it is consumed to break the deadlock cycle. Potential deadlocks can be detected with a time-out mechanism. If a packet is unable to make progress for a time duration corresponding to the

* *The research described in this paper was supported in part by an NSF Research Initiation Award, grant ECS-9411587, and by the Spanish CICYT, grant TIC94-0510-C02-01*

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

time-out, it presumes a potential deadlock situation and becomes eligible for recovery. A packet timing out does not necessarily imply deadlock; the time-out mechanism is simply sufficient guarantee that deadlock will never occur. *Disha* recovers from potential deadlocks regardless of the time-out value, provided that it is finite. Network bandwidth is allocated to deadlock recovery only in the rare instances when probable deadlocks are detected; otherwise, all network bandwidth is devoted to *true* fully adaptive routing of normal packets. Hence, routers designed in this fashion are not only simple and potentially faster, but also can result in enhanced communication efficiency and improved network performance. Exhaustive simulations confirm that this scheme is extremely efficient even when recovery is sequentially [2].

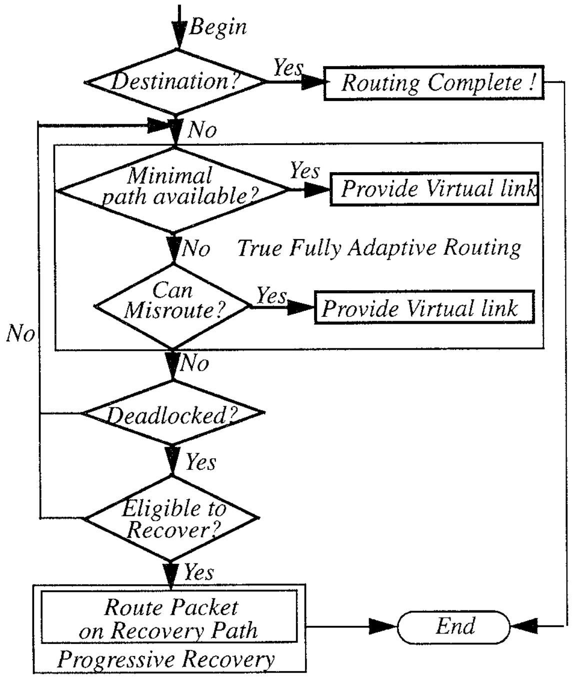

Figure 1: The *Disha* Routing Algorithm.

Disha is reminiscent of *Duato*'s algorithm [7] and Dally and Aoki's scheme [6] which use two virtual networks -- one susceptible to deadlocks (usually fully adaptive) and the other deadlock-free (possibly deterministic). However, there are significant differences. For one, escape paths in these schemes use edge buffers. Depending on the topology (mesh or torus) more than one virtual channel might be required. However, the escape channel in *Disha* is a single Deadlock Buffer central to the router, a nominal resource. This buffer is shared between neighboring nodes and, unlike edge buffers, is not dedicated to any one physical path. This allows existing virtual channels to be used solely for increasing performance, not for guarding against an infrequent event. Second, *Disha* is simpler and faster. It does not require packets to carry information about the number of dimension reversals and no comparison of the same is required.

The *Disha* algorithm as proposed in [2] requires packets to gain exclusive access to the Deadlock Buffers by capturing a circulating Token [1]. As recovery from cycles is sequential, the bandwidth available for recovery is small which could degrade performance if deadlocks are temporally clustered or otherwise appear more frequent (our simulations have not yet discovered such events).

In this paper we propose a new mechanism based on structured buffer routing that offers higher bandwidth for deadlock recovery on demand and eliminates the need for mutual exclusion on recovery resources. It does not have any single point of failure, and simultaneous recovery from multiple deadlock cycles is now possible with *Disha*. Because *Disha* Concurrent deadlock handling requires no supplementary virtual channels, router simplicity and speed is maintained. In this paper we also extend the theory developed in [7] to support a mixed set of resources using the notion of buffers (edge and central) as opposed to channels. This generalized theory is used to prove that *Disha* Concurrent recovery routing is deadlock-free.

The remainder of this paper is organized as follows. Section 2 presents our new theory for deadlock freedom based on the notion of buffers and applies it to *Disha* Concurrent. Section 3 gives performance results. Section 4 concludes with brief summary and scope for future work.

2.0 *Disha* Concurrent and Generalized Theorem for Deadlock Freedom:

2.1 Concurrent Recovery Methodology:

The simple methodology we use for designing *Disha* free from the requirement of mutually exclusive access to the deadlock-free lane is to structure the routing on the central Deadlock Buffers such that cyclic dependencies do not occur:

1) Construct a Hamiltonian path on the network using Deadlock Buffers.
2) Label the nodes in sequence along the Hamiltonian path (e.g., Figure 2 for a 2D mesh).
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 (not necessarily sequential), and

although we define a Hamiltonian path on the Deadlock Buffers, recovering packets can take shortcuts (e.g., from node 6 to node 15 in a single hop in Figure 2).

The Hamiltonian path as constructed above is equivalent to the high-channel network in [10]. This structured path allows a deadlocked packet to reach any node with a higher label. It is easy to see that the Hamiltonian path is acyclic and routing on the Deadlock Buffers is deadlock-free even when short-cut paths are allowed. Deadlocked packets at nodes higher than the destination label use normal edge buffers to reach an intermediate node lower in label than the destination, following which they use the acyclic Deadlock Buffer path to arrive at the destination.

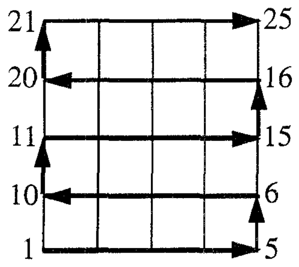

Figure 2: Hamiltonian path for a 5 ×5 mesh.

2.2 Generalized Theorem:

Results derived in [7] and [8] cannot be applied to schemes such as *Disha* which use both edge buffers (commonly referred to as virtual channels) and central buffers. As the routing function for edge buffers is usually different from that for central buffers, the routing function must consider the type of buffer in which the packet is stored at the current node. This paper develops a generalized version of the theory in [7] 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. As we only introduce changes in notation, our extensions do not change the way the theorem is proved which follows along similar lines as in [7]. 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 N represent the set of processing nodes and arcs C represent the set of communication channels connecting nodes. Each channel c_i has an associated queue or edge buffer. Additionally, each node n_j contains one or more central buffers accessible by all buffers at neighboring nodes. Let Q be the set of central buffers (i.e., Deadlock Buffers). Let b_k be an element of the set of buffer resources $B = C \cup Q$.

Definition 2:

An *adaptive routing function* $R : B \times N \longrightarrow \Pi(B)$, 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, $R(b_c, n_d) = \{b_1, b_2, ..., b_p\}$.

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. More precisely, $R_1(b, n) = R(b, n) \cap B_1, \forall b \in B, \forall n \in N$. 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.

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.

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 [7} and, therefore, not repeated here.

2.3 Application of Generalized Theorem to *Disha* Concurrent:

The generalized theory developed above 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. The idea is to augment the routing subfunction so that 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 proof of deadlock-freedom using the generalized theorem 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 from left to right. This labelling for a 5 $\times$ 5 mesh is illustrated in Figure 2.

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. 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 in a potential deadlock situation. The Deadlock Buffer q_j at node *j* can be used by a packet at any neighboring node for which the destination node d 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 be used only in increasing label order (shortcuts within this increasing order are allowed) and routing on the Deadlock Buffers is specified by the routing subfunction $R_{db} : B \times N \longrightarrow Q$ such that $R_{db}(b_j, d) = q_k$, where $\pi[q_k] = \max\{\pi[q_i] : \pi[q_i] \leq \pi[d]\}$ and q_i is a Deadlock Buffer at neighboring node *i*.

Assumption 5:

The routing function R_{min} which supplies the set of buffers C is fully adaptive and minimal. The overall routing function $R = R_{min} \cup R_{db}$. The transition from edge to Deadlock Buffers is handled by R_{db} and could follow a non-minimal path.

Constructing a Connected Routing Subfunction:

Let Q be the set of Deadlock Buffers. Only 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. It is formed by including an already existing edge buffer at every node. Each edge buffer has an associated unidirectional channel and, consequently, can be specified in terms of these channels. We include 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 3 for a 5 $\times$ 5 mesh. Let $B_1 = Q \cup C_{edge}$ be the buffers that define the routing subfunction R_1.

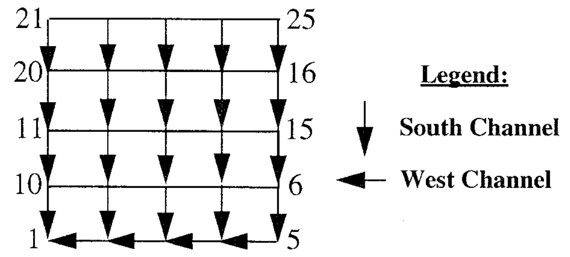

Figure 3: Channels added to make R_1 connected.

It should be noted that C_{edge} are existing edge buffers ($C_{edge} \subset C$) and not additional virtual 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 here).

Lemma 1:

The routing subfunction R_1 is connected.

Proof:

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 j such that $\pi[j] \leq \pi[d]$ using edge buffers C_{edge} (supplied by R_{min}) and the Deadlock Buffer at node j. The transition to the Deadlock Buffer at node j from an edge buffer at a neighbor of node j is governed by R_{db}. If $j \neq d$, a path from j to d can then be established using central buffers supplied by R_{db}. Thus, R_l is connected.

Lemma 2:

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

Proof:

It follows from Assumption 3 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 Deadlock Buffers in increasing label order. Thus, there are no cyclic dependencies between Deadlock Buffers. Hence, any cycle in the extended resource dependency graph of R_l cannot involve buffers belonging to Q.

Lemma 3:

The routing subfunction R_l 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, but none are cyclic. 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. There are no dependencies from South channels to South channels in higher rows, or from West channels to South channels. Because $B_l = Q \cup C_{edge}$, it follows that there are no cycles in the extended resource dependency graph. Figure 4 shows the extended resource dependency graph for C_{edge} of a 3×4 mesh; it is acyclic.

Theorem 2:

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

Proof:

From Lemma 1, the routing subfunction R_l 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.

■

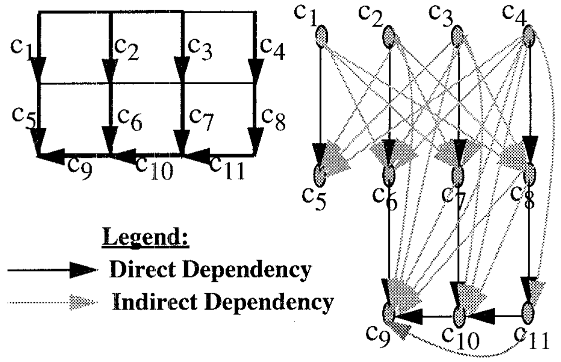

Figure 4: Extended resource dependency graph for a 3 × 4 mesh.

Concurrent Recovery for n-Dimensional Meshes and Toroids:

We consider the example of a 3D mesh; extending to any n-dimensional mesh is similar. A 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 5a. The path in a plane is similar to the one shown in Figure 2, 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 5b. All nodes in lower planes can be reached through the Deadlock Buffers (Q). To reach nodes in higher planes, the above Z^+, Y^- and X^- channels of C_{edge} can be used to reach the destination or the neighbor of some intermediate node with a lower label than the destination label. A path from this node to the destination can be established through Q. The routing subfunction defined on $B_l = Q \cup C_{edge}$ 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.

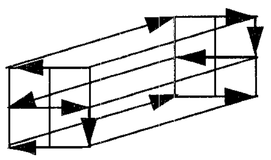

Figure 5a: Hamiltonian path formed by DBs.

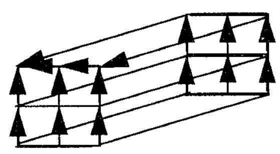

Figure 5b: Channels added to make R_l connected.

Theorem 3:

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

The torus requires two Deadlock Buffers at each node. We construct two distinct Hamiltonian paths using these buffers. Deadlocked packets use one path to reach higher label destinations and the other to reach lower label destinations. These Hamiltonian paths are acyclic even with short-cuts and again Theorem 1 can be applied to show that routing is deadlock-free.

Theorem 4:

"With two Deadlock Buffers, N-dimensional toroidal networks can safely recover from deadlocks when eligible packets are routed *concurrently* on them." (Proof in [3].)

3.0 Performance Results:

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

3.1 Comparisons for the Mesh:

Dimension ordered routing (*DOR*) is non-adaptive. *Duato* is based on the methodology proposed in [7]. *Disha-Seq* traces sequential recovery in *Disha*[1]. Concurrent recovery (*Disha-Con*) based on Section 2.3 is simulated for two different time-outs of 8 and 1000 cycles. The performance of *Duato* has been shown to be superior to other preventive schemes like Dally and Aoki and Planar Adaptive Routing [2]. Hence, we confine our comparison to *Duato* and *DOR*.

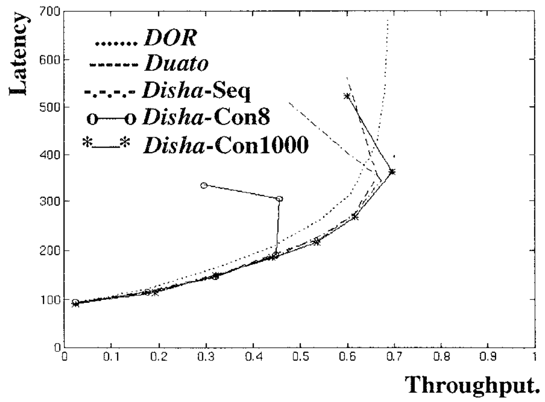

Fig. 6: Comparison for Uniform Traffic.

Under uniform traffic, performance is very similar for all schemes except *Disha-Con8,* as shown in Figure 6. The insufficient time-out in *Disha-Con8* causes false deadlocked (i.e., 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.*

We now compare the performance of our new algorithm for non-uniform traffic patterns. Performance results for the perfect shuffle traffic pattern given in Figure 7 shows a deterioration in performance for 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 [3]. Simulations for the torus show as much as 30 to 40% improvement in performance for *Disha* over *Duato.*

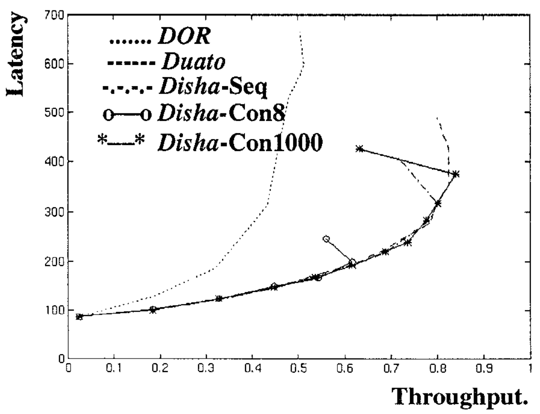

Fig. 7: Comparison for Perfect Shuffle traffic pattern.

3.2 Discussion:

Simulations indicate that, for concurrent recovery with *Disha*, superior performance is pivotal on accurate deadlock detection. All packets presumed deadlocked are eligible to recover and are routed on the Deadlock Buffer lane. 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 recovered. This is seen in the *Disha-Con8* performance results. Here, the time-out interval is insufficient to filter out blocked packets due to normal network congestion. The larger time-out with *Disha-Con1000* better filters out false deadlock detections, selectively allowing those packets truly in need of deadlock recovery to use the Deadlock buffer lane. Simulation runs for the mesh show that as much as 50% improvement in performance for *Disha* Concurrent is obtainable when deadlocks are detected more accurately. This suggests that more accurate deadlock detection mechanisms beyond that of time-outs may be warranted for concurrent recovery.

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. Mesh performance results of the *true*-fully-adaptive *Disha (Seq* and *Con)* recovery routing are almost similar to the fully adaptive *Duato* and far superior to the non-adaptive *DOR* deadlock avoidance schemes. For the torus, in *Duato*, the choice of virtual channels at a node for unrestricted routing is reduced. This explains the vast improvement in performance for *Disha* [3]. Similar results are expected as we go to higher dimension networks. It should be noted that our new approach functions without the need for a Token resource. Hence, the improved performance comes with a more reliable, less costly implementation.

4.0 Conclusions and Future Work:

Deadlock avoidance schemes constantly share physical bandwidth between virtual channels used both for deadlock avoidance and routing. The main advantage of recovery is that no bandwidth is "wasted" on avoiding situations that generally occur infrequently. 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 on demand to deliver deadlocked packets progressively. In the absence of deadlocks, all channels (physical and virtual) are fully devoted to normal packet routing. Once deadlock is detected, *Disha* de-allocates physical channel bandwidth from some packets and reassigns it to others to enable recovery. Unlike virtual channels, Deadlock Buffers used in recovery are not associated with any specific physical channel, and network bandwidth is allocated to recovery only on rare deadlock events. This results in increased sharing of network resources for enhanced communication efficiency and improved performance as confirmed by simulations.

Previous proposals for *Disha* were based on sequential recovery. Here, we propose a first extension that allows concurrent deadlock recovery without the need for a special Token resource. No supplementary buffer resources are required over the original scheme for n-dimensional meshes. N-dimensional toroids require no more than one additional central flit-buffer. Another significant contribution of this paper is the development of a generalized theory for deadlock freedom applicable to both deadlock avoidance and deadlock recovery schemes. The extension is developed using the concept of buffers as opposed to channels. This generalized theory is used to prove that the recovery based *Disha* Concurrent scheme 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 just a single Deadlock Buffer per router. Other interesting areas to explore are methods to increase the fault-tolerance capabilities of *Disha* so that, in the presence of faults, there is at least one alternate path from a node to any other node.

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*, 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*, pages 201-210, June 1995.

[3] Anjan K.V., Timothy M. Pinkston, and José Duato. Deadlock-Free Adaptive Wormhole Routing with *Disha* Concurrent. *USC CENG Technical Report*, October 1995.

[4] 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*, pages 268-77, May, 1992.

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

[6] 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.

[7] 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.

[8] J. Duato. A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks. In *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1055-1067, October 1995.

[9] 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*, pages 289-300, April 1994.

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

[11] 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.

[12] L. Ni and C. Glass. The Turn Model for Adaptive Routing. In *Proceedings of the 19th International Symposium on Computer Architecture*,20(2):278-287, May 1995.