References:

[1] Anjan K. V. and Timothy Mark Pinkston. *DISHA*: An Efficient, Fully Adaptive Deadlock Recovery Scheme. *CENG Technical Report 94-23, Department of Electrical Engineering - Systems, University of Southern California, Los Angeles, CA 90089-2562*, November 1994.

[2] 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.

[3] 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.

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

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

[8] 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.

[9] J. Duato. On the design of deadlock-free adaptive routing algorithms for multicomputers: design methodologies. In *Proceedings of Parallel Architectures and Languages Europe,* pages 390-405, June, 1991.

[10] 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.

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

[12] M. R. Feldman, et al. Comparison between Optical and Electrical Interconnects based on Power and Speed Considerations. In *Journal of Applied Optics*, Vol. 27, No. 9, pages 1742-1751, May 1988.

[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] Z. Liu and A. Chien. Hierarchial Adaptive Routing: A Framework for Fully Adaptive and Deadlock-Free Wormhole Routing. In Proceedings of the *Symposium on Parallel and Distributed Processing*, 1994.

[16] 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.

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

[18] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks 3 - North Holland Publishing Company,* pages 267-286, 1979.

[19] Timothy Mark Pinkston. The GLORI Strategy for Multiprocessors: Integrating Optics into the Interconnect Architecture. *Ph.D. Thesis: CSL-TR-92-552, Stanford University,* Stanford, California 94305-4055, Pages 16-19, December 1992.

[20] D. C. Wong et al. A Bipolar Population Counter using Wave Pipelining to achieve 2.5xNormal Clock Frequency. *IEEE Journal of Solid State Circuits, Vol. 27, No. 5*, pages 745-753, May 1992.

4.1 Uniform Traffic:

Figure 8 compares *Disha* with preventive schemes under uniform traffic and four virtual channels. Equal clock cycle time was assumed for all schemes. *Disha* was simulated with a time-out of eight clock cycles. *Disha* outperforms all schemes and more than doubles the saturation load. *Disha* saturates at 0.65, and dimension-order a distant second at 0.3. The fact that dimension-order outperforms partially adaptive schemes is not unexpected as it preserves the traffic's uniformity (others [15] have recorded similar results). However, the performance of dimension-order relative to the other schemes is expected to deteriorate as the number of dimensions increase.

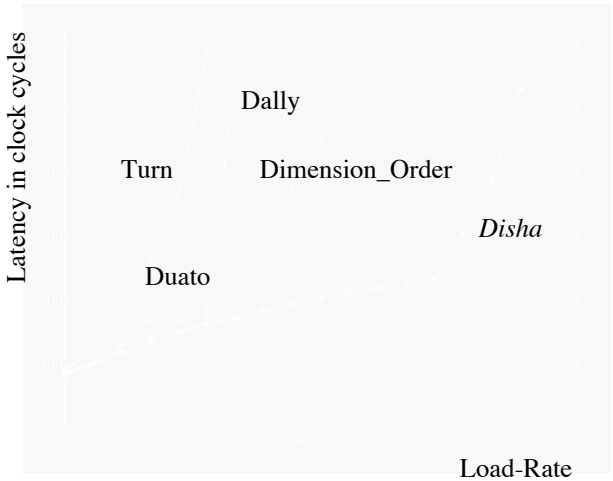


Fig. 8: Comparison for Uniform Traffic Pattern and four virtual channels.

4.2 Non-Uniform Traffic:

A comparison is also made for hot spot traffic conditions (Figure 9). In these simulations, we assumed that up to 5% of the network traffic is hot spot in nature, and the number of locations at which these patterns exist is one. This implies that 5% of all network traffic is trying to reach one destination that is randomly selected. These simulations were run with the same time-out value as for uniform loads. We expect the performance to improve with fine tuning in the form of increasing time-outs to filter out false detections. Hot spot traffic causes early saturation for all schemes. Nevertheless, *Disha* provides the best results. The latency after saturation rises more steeply for the preventive schemes as compared to *Disha*.

7.0 Conclusions and Future Work:

Disha provides the means for safely incorporating fully adaptive wormhole routing to support efficient and fault-tolerant communication in a parallel processing

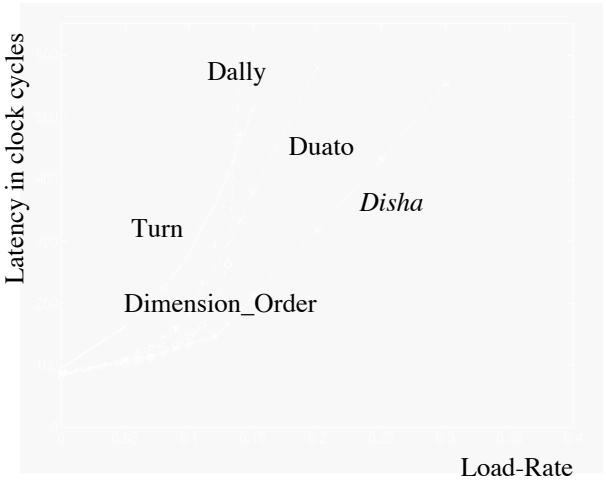


Fig. 9: Comparison for hot-spot traffic pattern, Hot Spot = 5% to one node.

environment. The scheme is universal in that it can be applied to any arbitrary network topology. It employs deadlock recovery as opposed to prevention with the objective of making the common case faster. Consequently, *Disha* does *not require* "edge" buffers for deadlock freedom. All virtual channels are used efficiently for fully adaptive routing. Moreover, hardware devoted to recovery is minimal, enabling the routers to be fast. This novel idea proves to be very effective as confirmed by exhaustive simulation [1].

There are many areas of future research resulting from this work. Without major changes to the basic scheme, *Disha* could be extended to support two Deadlock Buffers as an escape scheme to allow simultaneous recovery from a number of deadlock situations or as an implementation technique in lieu of our asynchronous token-passing protocol. This should further improve performance and fault-tolerance. Reducing the probability of deadlocks is another related area of interest. Finally, a complete study on the characterization of deadlocks is necessary to extract maximum performance from a recovery scheme such as *Disha*. For instance, the optimum time-out value depends on message length, buffer depth, traffic pattern, topology, etc. Such a study will help in fine-tuning the system to obtain peak performance.

Acknowledgments:

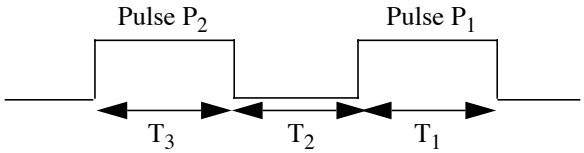
The authors are extremely grateful to Binh Vien Dao at Georgia Institute of Technology for helping us set up Flitsim 2.0, which was used in all the simulations presented in this paper. Enlightening discussions with Jose Duato have proved invaluable. Helpful suggestions from Sudhakar Yalamanchili were also very useful.

falling edge of the Token. When the Token comes around the next time, it is captured and sunk by only one router. This protocol ensures mutual exclusion.

One possible implementation of the Token as a propagating pulse requires just two gates on the critical path including a gate to propagate a regenerated Token pulse [1]. The Token can therefore be clocked much faster than the router to enable quick deadlock recovery initiation. The Token seizure time can thus be reduced considerably from the case where the Token and the router clock are in synchronism.

A disadvantage with this protocol is that the routers have to wait a full Token cycle before they can seize it. The average Token capture time is now given by $T_pN/2 + T_pN = T_p3N/2$. For N = 256 and assuming that T_p is 1/5 of the router clock period, the average Token capture time is about 75 clock cycles. This reduction in the Token capture time from 128 clock cycles justifies any increase in Token implementation complexity.

The Token capture time can be further reduced by another factor of three with a clever improvisation.



A router sees pulse P_1 before pulse P_2.

Fig. 6 A two pulse-train Token.

The Token is modified to have two pulses as shown in Figure 6. The DBit is always set at the falling edge of pulse P_1. P_1 thus effectively acts as the synchronizer. Pulse P_1 would go onto the next router R_2. If router R_1 is in a deadlocked state, then pulse P_2 does not make it to R_2. Router R_2 could synchronize with pulse P_1 setting its DBit. However, if it does not receive pulse P_2 in the specified time interval T_2, it resets its DBit. The router regenerating the Token issues both pulses P_1 and P_2. The average Token seizure time is now reduced to $T_pN/2 + T_1 + T_2$ which is approximately 25 clock cycles. Given that there is more than just a single packet involved in deadlock cycles, the seizure time for deadlock recovery initiation will certainly be further reduced. Moreover, this latency for Token capture can be hidden by taking it into account in the time-out.

Considering Token regeneration, the Token cannot simply be released by the node that captures the Token once the *tail* flit has gone by. The *tail* could have gone by long before the packet reaches its destination. This would be particularly true if we had small packets with deep networks (large buffer sizes) or networks with large diameters. Another node could potentially seize the Token and mutual exclusion on the Deadlock Buffers would be violated. To avoid this, we instead allow the *destination* to release the Token once it receives the *header*. A packet once placed on the Deadlock Buffer continues to follow the Deadlock Buffer path until it reaches its destination. The destination node releases (regenerates) a new Token if it receives a packet on this path.

4.0 Performance:

This section of the paper evaluates the performance of *Disha* using a modified version of FLITSIM 2.0 [†] . All simulations are run on a 16 by 16 two dimensional torus. Messages are 32 flits long. A buffer depth of two was selected (shallow buffers keep the routers simple and reduce clock cycle time). All algorithms simulated used one injection and reception channel per node.

This paper is premised on the notion that deadlocks are rare. Figure 7 verifies this for two widely varying time-out thresholds, 4 and 64. Load-Rate here and in the remainder of the graphs is a fraction of full load, defined as the load at which all channels in the network are used simultaneously (maximum network capacity). The figure shows the number of packets that seized the Token normalized with respect to the number of packets that were delivered by the network. The simulations were done for three virtual channels under uniform traffic. The number of deadlocks are closely related to the number of packets that seize the Token. Based on these results, it is safe to infer that deadlocks are extremely rare up to the point at which the network saturates (*Disha* with three virtual channels saturates at a load-rate of approximately 0.40). This confirms earlier measurements by Kim, et al. [14], which showed that deadlock situations are generally rare.

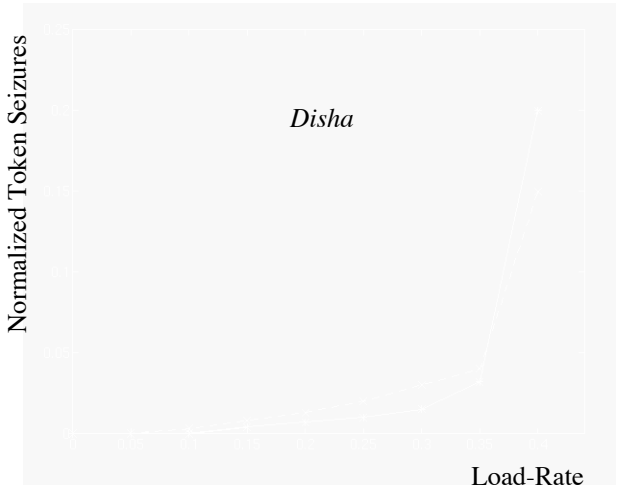
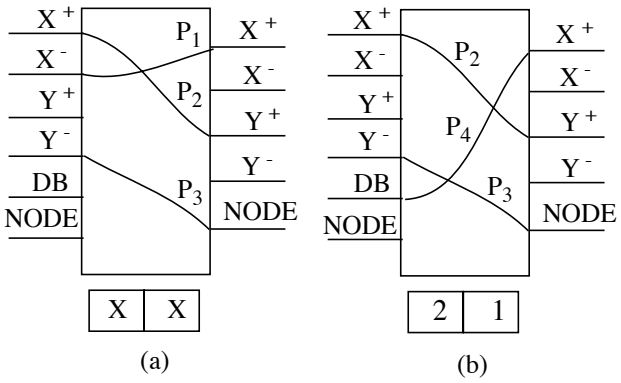


Fig. 7: Normalized number of packets that seized the Token.

[†] FLITSIM was developed by Patrick Gaughan (currently at the Univ. of Alabama) et al,. at Georgia Inst. of Tech.



2 | 1 This notation implies that input 2 (X^-) was connected to output 1 (X^+).

Fig. 3 shows the reconfiguration of the crossbar.

We cannot do without reconfiguration because with the Token held at this router, packet P_1 may be blocked further upstream (and never clear) leading to a deadlock configuration.

3.0 Implementing Mutual Exclusion on the Deadlock Buffer Lane:

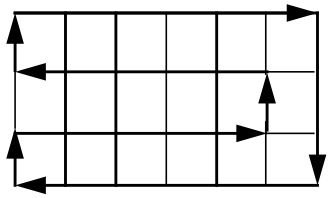


Fig. 4 Hardwired Token path.

This section discusses some details of the simplistic logic needed for implementing an asynchronous token-passing protocol. Figure 4 shows one possible hardwired Token path (a Hamiltonian cycle) for a mesh or torus. The cost of the Token line is about '$1/k^{th}$' of a regular status line as it does not include all paths in the network, where 'k' is the node degree. The Deadlock Bit (DBit) of a router indicates whether or not the Token should be captured. As will be explained later, the DBit is not necessarily set as soon as deadlock is detected.

The Token can be clocked in synchronism with the router clock. This, however, restricts the Token to propagate at the router speed. On average, a node would then have to wait N/2 clock cycles to capture the Token, where 'N' is the network size (number of nodes). For a 256 node network, the average time to capture the Token would then be an unacceptable 128 clock cycles.

Routers propagate an input Token, unless the DBit has been set. The Token implementation minimally requires just one gate on the critical path. Hence, it would be profitable to implement the Token as an asynchronous signal (with respect to the router clock). The Token is now defined as a propagating pulse, where PW is the pulse width and T_p is the propagation delay from router R_1 to R_2. T_p includes the gate delay at a router and the wire delay between routers. (The effect of wire delay may be reduced with wave-pipelined transmission [20] or free-space optical interconnects [12,19]). To ensure that the Token can be safely latched-in when required, the pulse width should minimally be approximately two and a half times the latch clock period (router clock). If this were not the case a situation could arise where the edge of the clock misses the Token.

Since the Token is a propagating wide-band pulse, we now have a situation in which the Token could be simultaneously present in two or more routers at the same time, resulting in a mutual exclusion problem. With the Token running asynchronously with respect to the router clock, it is possible that router R_1 changes the state of the DBit at an instant when the Token is passing through it. Meanwhile, the initial part of the Token has already propagated to router R_2. This router R_2 could already be in a deadlocked state with its DBit being set even before the Token arrives. Now both routers R_1 and R_2 could potentially seize the Token as shown in the timing diagram below (Figure 5).

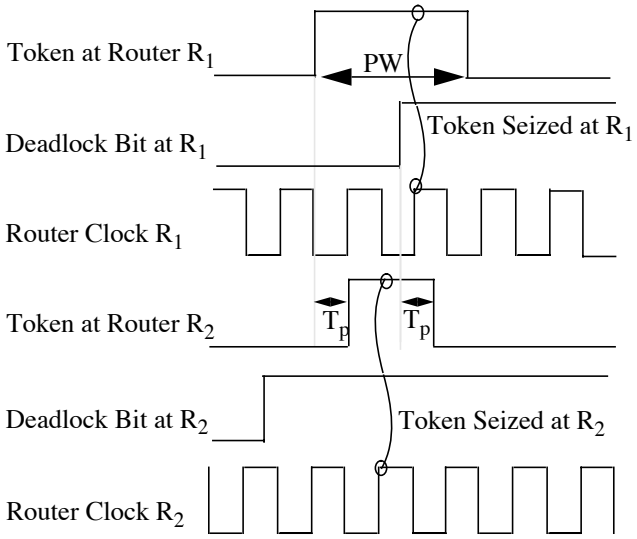


Fig. 5 Illustrates how two routers could potentially seize the Token at the same time.

The problem could be solved if router R_2 were to detect the reduction in pulse width and allow R_1 to seize the Token. Since it would be difficult to implement this, we propose a different solution.

A router is prohibited from changing the state of the DBit while the Token is passing through it. The routers follow a protocol by which they are allowed to set the DBit at the

P_2, are able to proceed. Recovery is achieved. It should be noted that allowing any one of the packets to proceed breaks the entire deadlock cycle. There is no ordering as to which packet should be placed on the deadlock-free lane.

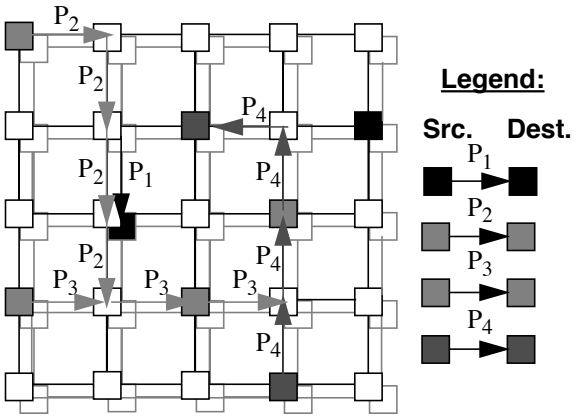


Fig.1 shows deadlock recovery with *Disha*.

One possible implementation of the required mutual exclusion on the Deadlock Buffer path is to have a Token to circulate in the network along a fixed predetermined cyclic path to include all routers. We discuss this in further detail in Section 3. Figure 2 below shows why mutual exclusion is necessary and how a deadlock situation could arise even under deterministic routing on the special buffer path. Even though P_1 is being routed along the y - dimension, it still waits on P_2 which is being routed along the x - dimension as P_2 has occupied the Deadlock Buffer of that particular router. Likewise P_2 -> P_3 ->P_4 -> P_1, leading to a dependency cycle on the special buffers.

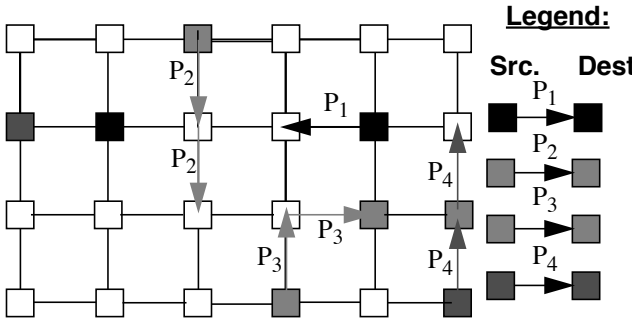


Fig. 2 shows how deadlocks could occur on the special buffer path under x-y routing.

A detailed proof that *Disha* is deadlock free can be found in [1]. Here, we give only an intuitive proof of the scheme. Assume that the network is deadlocked. There may be one or more dependency cycles. Assume that there are 'k' dependency cycles. A packet can be placed on the cycle-free Deadlock Buffer path. Following this path, the packet is guaranteed to reach its destination where it will be sunk. This eliminates one cycle, reducing the number of cycles to (k-1). By induction, it follows that the network safely recovers from deadlocks.

2.1 Deadlock Detection and Handling:

Deadlocks can be detected using a time-dependent selection function. The idea is similar to those suggested in [13] and [14]. The router waits for the arrival of a packet. On receiving one, it resets $T_{elapsed}$, which keeps track of the number of clock cycles this router is unable to send out the header. If unable to send out the header, $T_{elapsed}$ is incremented. The router continues to increment $T_{elapsed}$ on every cycle until it is successful in sending out the *header* or the time-out (T_{out}) interval for that packet has been reached. When $T_{elapsed} > T_{out}$, the router changes its state to "deadlocked". With just a single "central" buffer per node, the routers need to obtain exclusive access to the special buffer path before they are allowed to place a packet on it (exclusive access would not be required if we had more than just a single Deadlock Buffer). Once exclusive access has been obtained (i.e., by capturing a token), a router places the deadlocked packet on the output channel with the corresponding status line asserted to indicate that it should be placed in the Deadlock Buffer at the next router. It simultaneously changes its status to indicate that it is no longer "deadlocked". The packet is guaranteed to reach its destination following the Deadlock Buffer path.

2.2 Deadlock Recovery:

In the absence of data on the Deadlock Buffer, the router's crossbar is configured as specified by the decision and arbitration logic. However, under a deadlock situation, it could so happen that the Deadlock Buffer input to the crossbar is required to be connected to the same output as is currently being used by some other packet on the "edge" input buffers. The crossbar would then have to be reconfigured to connect the Deadlock Buffer to the required output terminal. The decision logic thus needs to remember the state of the crossbar before it was reconfigured so that it can be reconnected once deadlock has cleared.

To illustrate this point, Figure 3a shows the initial crossbar configuration before a deadlock situation. With deadlock and data requiring to be put out on X^+, the crossbar is reconfigured as shown in Figure 3b. The reconfiguration buffer stores the state of the crossbar before change-over. This buffer does not need to store the entire crossbar state, but only that of the input that was disconnected. Data on the Deadlock Buffer is guaranteed to reach its destination and once packet P_4 has cleared, the crossbar is returned to its initial configuration (Figure 3a) using the information stored in the reconfiguration buffer.

even oblivious routing algorithms.

Deadlock recovery is an alternative to prevention. Simulation studies in [14] and Section 4 of this paper show that deadlocks are typically infrequent. Recovery thus seems to make more sense than prevention. Compressionless Routing [14] is an effort in this direction. In contrast to other adaptive schemes (with the exception of Turn Model), it does not require multiple virtual channels. Compressionless Routing detects potential deadlock situations and recovers from them. This approach gives better performance in the common case. This scheme, however, requires padding which decreases effective channel utilization -- when the packet sizes are small compared to the network diameter or the buffers at routers are deep, the overhead due to padding is large. Also, packets have to be stored to allow retransmission if necessary, and killed packets suffer increased latencies. Moreover, injector and receiver interfaces, additional counters, and multiple status lines increase the hardware complexity.

We believe that routing should be fully adaptive. If deadlock occurrences are extremely rare then it does not make sense to limit the adaptivity of the routing scheme to solve an infrequent event. Limiting adaptivity also reduces fault tolerance capabilities of the system. We further believe that the router design should be extremely simple. It does not make sense to devote virtual channels specifically to prevent deadlocks; virtual channels should be used primarily as a means of improving flow control [7]. The goal here is to make the common case fast and *Disha* is a solution to this.

Disha provides the framework for supporting fully adaptive wormhole routing. Although the examples considered here are mesh type, this technique can be applied to any interconnection network topology. It is a deadlock recovery strategy and, consequently, does not require virtual channels for prevention. It offers the following advantages: 1) deadlock-free fully adaptive wormhole routing, 2) applicability to arbitrary interconnection network topologies, 3) no virtual channels devoted to the prevention of deadlocks, 4) simple router design, and 5) no padding, storing or retransmitting of packets required.

This paper discusses a simple method for detecting and quickly recovering from deadlock situations. The token-passing protocol featured in this paper is only one of several methods for implementing *Disha*. The remainder of the paper is organized as follows: Section 2 gives an overview of the *Disha* recovery scheme. Section 3 discusses the asynchronous token-passing protocol which implements mutual exclusive access to the deadlock lane. Section 4 presents some simulation and performance results. Finally, Section 5 gives our conclusions and future work.

2.0 *DISHA* Recovery Scheme:

Disha [†] aims at optimizing routing performance in the absence of deadlocks. This is achieved by allowing routing to be fully adaptive over all virtual channels and dealing with the rare cases when deadlock does occur. This scheme requires nominal hardware to be devoted to deadlock recovery.

Each router is provided with an additional special flit buffer (Deadlock Buffer) to be used in the case of deadlocks. These Deadlock Buffers are central to the routers and essentially form a dedicated deadlock-free lane which can be viewed as a "floating" virtual channel. Routing on this deadlock-free lane can be fully adaptive. 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. To ensure that the special buffer path is deadlock free, only one packet should be allowed to use it at any given time. Section 3 is devoted to addressing this issue.

Disha is consistent with Duato's scheme [9] of having two virtual networks - one susceptible to deadlocks (possibly adaptive) and the other deadlock free (possibly deterministic). However, there is a significant difference. *Disha* uses "central" Deadlock Buffers that form a deadlock-free lane. These buffers are essentially input buffers. Virtual channels ("edge buffers") are not devoted exclusively for this purpose. The commandeering of output edge buffers or physical channel resources (in the absence of output buffers) for deadlock recovery will be occasional given that the number of deadlock occurrences are infrequent. Hence, fully adaptive routing on all the provided virtual channels is permitted all of the time and should result in significant performance gains.The key idea behind this is that the cost complexity of the virtual channel controller (VCC) with each additional virtual channel grows at a much faster rate as compared to that of the crossbar with each additional input, as pointed out by Chien [5]. Because *Disha* does not require virtual channels (edge buffers which grow with network dimensionality) for deadlock freedom and has nominal increase in crossbar complexity (due to a singular central deadlock buffer per router), it should outperform preventive schemes which do require virtual channels and have large router crossbars as a result.

To illustrate how a potential deadlock situation can be recovered from, we provide the following example. In Figure 1, the special buffer path can be viewed as an alternate network interleaved with the original. Using this alternate path, a packet, say P_1, upon detection that it is deadlocked, can reach its destination where it will be sunk. This breaks the deadlock cycle and other packets, P_4, P_3 and

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

In Proceedings of *The 9th International Parallel Processing Symposium,*
April 24-29, 1995 at Santa Barbara, California.

DISHA: A Deadlock Recovery Scheme for Fully Adaptive Routing *

Anjan K. V. Timothy Mark Pinkston

Electrical Engineering - Systems Department, University of Southern California
3740 McClintock Avenue, Los Angeles, CA - 90089 - 2562
{anjan@truth, tpink@charity}.usc.edu; http://www.usc.edu/dept/ceng/faculty.html/pinkston/home.html

Abstract:

This paper presents a simple method of implementing an efficient and cost effective routing scheme. The strategy considers deadlock recovery as opposed to prevention to optimize performance in the absence of deadlocks. Cycles are broken by re-routing a blocked packet through a deadlock-free lane which is implemented as a central "floating" buffer. The proposed scheme is extremely simple, ensuring quick recovery from deadlocks and enabling the design of fast routers.

1.0 Introduction:

The interconnection network is the backbone for communication in a multiprocessor/multi-computer environment. System performance is determined not only by the effective utilization of multiple processor nodes but also, to a large extent, by efficient communication amongst the nodes. For this reason, many schemes which incorporate wormhole switching [17] and adaptive routing [13] have been proposed to increase communication efficiency.

Most wormhole adaptive schemes focus on deadlock prevention rather than deadlock recovery. Preventive schemes suffer from high hardware complexity and/or losses in adaptivity. Considering first how adaptivity can suffer, we examine Dally and Aoki's proposed *dynamic routing algorithm* [8]. Here, every physical link is split into (r+1) virtual channels. Virtual channels are used to break deadlock dependency cycles. Each packet keeps track of the number of dimension reversals it suffers. A packet, when blocked with all suitable output channels being currently used by packets with equal or lower values of dimension reversals, is routed deterministically. Although this scheme prevents deadlocks from occurring, the number of virtual channels and the number of dimension reversals ultimately place an upper bound on the algorithm's adaptivity.

Considering hardware complexity, a recent study by Chien [5] shows that the increased complexity of virtual channel controllers adversely affects the machine performance by slowing down the router's clock cycle. Because of this fact, deterministic schemes might, in fact, outperform certain adaptive schemes. To address this problem, Chien and Kim proposed a partially adaptive scheme, called Planar-Adaptive Routing [4], which reduces the hardware complexity for deadlock prevention. The resulting increase in router speed, however, comes at a cost of restricting the degree of adaptivity to only two dimensions at a time.

Duato's algorithm [9], is a more flexible adaptive routing scheme that ensures deadlock prevention. Virtual channels are split into two groups -- adaptive channels and escape channels. Packets can be routed adaptively using any available adaptive virtual channel. If blocked, they are routed deterministically using the escape channels. Packets on the escape channels can come back onto the adaptive channels if they happen to be free at a subsequent router. The necessary and sufficient condition presented in [10] relaxes the situation even further by giving more flexibility to the packets on the deterministic channels.

Although the number of virtual channels (and the concomitant implementation complexity) required by Duato's scheme can be small, this scheme still has some disadvantages. For example, using Chien's model [5], if one determines that the optimum number of virtual channels for the expected load conditions is three, then for a torus type mesh architecture, Duato's scheme can use only one virtual channel for adaptive routing. Chien [5] goes on to show that, due to increased router latency, every additional virtual channel would require a 30 - 50% increase in load to justify its presence. This being the case, it seems unjustified to devote some portion of the virtual channels exclusively for deadlock prevention via deterministic routing.

The Turn Model [16] is a partially adaptive deadlock prevention scheme that does not require virtual channels. Freedom from deadlock is ensured by prohibiting certain turns in the routing algorithm. However, simulation studies by Boppana and Chalasani [3] show that this scheme (which is not applicable to arbitrary network topologies) produces unbalanced traffic conditions and can be outperformed by

* *The research described in this paper was supported in part by an NSF Research Initiation Award, grant ECS - 9411587, and by a grant from the Zumberg Fund and the Powell Fund.*