Deadlock Avoidance vs. Deadlock Recovery

Matthew Eldridge Peter Mattson John Owens

May 10, 1999

1 Introduction

Interconnection networks distribute packets between nodes of a network. Without careful consideration, some combination of these packets could interact in such a way to cause *deadlock*, defined as a situation in which "every packet is requesting resources held by other packet(s) while holding resources requested by other packet(s)." [4]

There are several techniques for coping with the problem of deadlock. One general category of these techniques involves avoiding deadlock altogether. *Deadlock avoidance* structures the algorithms used to route packets through the network to guarantee that a deadlock situation will never occur.

Another approach is *deadlock recovery*, in which deadlock is permitted to occur; the network must be able to first, detect deadlock, and second, eliminate the deadlock.

In this paper, we analyze examples of both deadlock avoidance and deadlock recovery. Loren Schwiebert and D. N. Jayasimha gave a compelling argument for avoiding deadlock in their 1995 paper [7]. Two years later, Timothy Pinkston and Sugath Warnakulasuriya presented a case [6] for their deadlock recovery architecture.

2 Deadlock Avoidance

The case for deadlock avoidance is simple. By stopping deadlock from ever occurring, the network does not have to cope with the difficulties of recovering from deadlock.

However, structuring the routing algorithms to guarantee that deadlock will never occur necessarily restricts the routing algorithms, which reduces performance by reducing path diversity.

The challenge for deadlock avoidance methods, then, is to develop a theory of deadlock, and from this theory, routing algorithms that require the least restrictions on routing.

Earlier proof techniques provided sufficient, but not necessary, conditions to avoid deadlock. José Duato presented one of the most important advances in this subject in [3], which "develops the theoretical background for the design of deadlock-free adaptive routing algorithms for wormhole networks." Duato's theory, in turn, builds on the work of Dally and Seitz [2], which showed that a nonadaptive wormhole routing algorithm was deadlock-free if and only if the channels were acyclic.

Schwiebert and Jayashima make two contributions to the theory of deadlock avoidance:

- They prove a necessary and sufficient set of conditions for deadlock freedom.
- From these conditions, they propose a routing algorithm for hypercubes that outperforms previous algorithms.

Theory Most techniques for deadlock avoidance allow only acyclic buffer usage, or equivalently, prohibit a cyclic buffer ordering. This paper argues that deadlock is only a danger when *waiting buffers*, defined as "buffer(s) that a packet can wait to acquire when all the output buffers permitted by the routing relation are unavailable," constitute the only resources for which a cycle is prohibited, as opposed to the set of all buffers. They point out that a deadlock-free routing algorithm could allow a packet to use a buffer only if that buffer is empty (and hence not a waiting buffer), and not allow the packet to wait for the buffer if it is full.

In their necessary and sufficient condition proof, they use the concept of a Buffer Waiting Graph (BWG) whose edges are "pairs of buffers (q_1, q_2) where a packet that currently occupies q_1 can wait for buffer q_2". A BWG can have both "true" and "false" resource cycles, which differ in that false cycles are not physically possible because they require at least one buffer to be used simultaneously by multiple packets to create the cycle.

They also define a routing algorithm as *wait-connected* if a packet always has a waiting buffer. In the proof, they demonstrate that a routing algorithm is deadlock-free iff it is wait-connected and its BWG is acyclic. They also prove the case that a routing algorithm is also deadlock-free even if it requires a blocked packet to wait for a specific output buffer iff it is wait-connected and its BWG has no true cycles. Finally, a routing algorithms which allows blocked packets to wait for multiple output buffers is also deadlock-free iff we modify its BWG by removing edges until all true cycles are resolved and the routing algorithm is wait-connected on this new BWG.

Results From these more liberal conditions for deadlock avoidance, Schwiebert and Jayashima propose an "Enhanced Fully Adaptive Routing Algorithm", a routing algorithm for hypercubes which is more adaptive than previous algorithms. Previous algorithms required one set of buffers to be used for only nonadaptive routing; dimension-ordered routing, for instance. This new algorithm allows partially adaptive routing on this set of buffers.

Over an n-dimension hypercube, they compare their "Enhanced Fully Adaptive" algorithm to an e-cube (dimension-ordered) routing and to Duato's algorithm. They quantify their improvement with the metric "Degree of Adaptiveness", defined by Glass and Ni in [5] as "the ratio of the number of paths permitted by the routing algorithm to the total number of paths, averaged over all source-destination pairs". Over all n studied ($2 \leq n \leq 12$), the enhanced algorithm outperforms Duato's algorithm, which in turn outperforms the e-cube routing. At $n = 12$, the enhanced algorithm still has a degree of adaptiveness of over 50%, while Duato's implementation only scores 16%.

3 Deadlock Recovery

In contrast, the theory of deadlock recovery takes the opposite view: Deadlock happens infrequently and instead of hobbling routing algorithms to completely avoid it, allow it to occur and concentrate your effort on removing the deadlock when it does. Pinkston and Warnakulasuriya characterize this distinction as a tradeoff between routing freedom and deadlock formation. The two tasks inherent in deadlock recovery are first, deadlock detection, and second, recovery from a deadlock condition.

In concurrent work ([8]), the authors contend that deadlock recovery is *viable* because deadlocks are rare. Given this, they explore the following topics:

- They model network traffic using a network simulator to acquire an understanding of deadlock formation given varying network parameters, such as routing adaptivity, virtual channel count, and buffer configurations;
- they use this information to compare deadlock recovery methods to deadlock avoidance methods; and
- they explore how networks can best be designed to take advantage of deadlock recovery algorithms.

Theory Their previous work ([8, 9]) characterizes classes of deadlocks: single-cycle, multi-cycle, and cyclic non-deadlocks. Having established this, they then turn to factors that influence deadlock formation probability, in particular *routing freedom*, the number of *blocked messages*, and the number of *resource dependencies*.

Routing freedom increases with the number of available physical channels and the number of virtual channels per physical channel, as well as the flexibility of the routing algorithm. Similarly, blocked message capacity scales with physical channel count and VC/PC count; it also increases with the addition of more buffer space per message (by either increasing overall buffer space or decreasing message size).

They then provide an analysis demonstrating that increased routing freedom, while exponentially increasing the *potential* for cycles[1], exponentially decreases both the number of blocked messages and the probability of deadlock.

[1] Many of these cycles do not lead to deadlock; i.e., they are cyclic non-deadlocks.

Simulation To back up this theoretical analysis, the authors develop and use a flit-level simulator called *FlexSim* on k-ary n-cube networks, and in particular, a 16-ary 2-cube with random traffic, a 32-flit fixed message size, a 2-flit buffer size, and a minimal "true fully adaptive routing" (TFAR) algorithm in which any profitable channel can be taken. They vary load rate, the number of virtual channels per physical channel, and the routing algorithm (TFAR vs. dimension ordered routing [DOR]) and measure deadlock count. The simulation can detect deadlock but makes no effort to repair it, instead deleting deadlocked messages until the deadlock is clear.

They first examine DOR and find that DOR with three or more virtual channels never deadlocks in practice. DOR1 (DOR with one virtual channel) deadlocks appear at lower load rates than DOR2 (DOR with two virtual channels) deadlocks[2]. TFAR, on the other hand, only deadlocks in practice with one virtual channel and does not deadlock with two channels, indicating that additional routing freedom is instrumental in avoiding deadlock. They then instead study precipitating factors for deadlock: message blocking and cycle formation, and find that at network saturation, cyclic non-deadlocks begin to grow rapidly, thus degrading overall router performance. They also analyze the tradeoff of deeper buffers against more virtual channels and find that, from a deadlock perspective, using many shallow virtual channels avoids deadlock better than using few deep channels.

The final part of this paper compares a deadlock avoidance routing algorithm (Duato, based on [3]) against Anjan and Pinkston's deadlock recovery architecture, Disha [1]. Because Duato restricts routing in the first two virtual channels to avoid deadlock and Disha does not, Disha's performance is superior for the same number of virtual channels, although this lead diminishes with each additional virtual channel added. Duato is also susceptible to cycle formation at lower loads for the same number of virtual channels, although at high loads Disha is susceptible to a greater number of cycles. From these results, they conclude that deadlock recovery based schemes are superior to avoidance schemes if sufficient routing freedome makes deadlock highly improbable.

4 Analysis and Conclusions

Both deadlock avoidance and deadlock recovery methods usually have a set of special channels distinct from the normal (and most efficient) routing channels that visit every node. Deadlock avoidance implementations usually have one or more channels on which the routing is restricted to prevent potential deadlock situations from forming; deadlock recovery implementations typically have an escape channel from which a deadlocked set of packets can break a cycle.

At this level, deadlock avoidance schemes and deadlock recovery schemes differ in only one salient point: in deadlock avoidance, this additional family

[2]Frankly, it's unclear from the paper how DOR deadlocks at all — it mentions that "no virtual channel class ordering is enforced", which evidently indicates that deadlock is possible; perhaps the cycle is formed from the wraparound in the torus.

of channels is used in normal routing, whereas in deadlock recovery, the extra channels are only used in exceptional circumstances.

Criticisms Schwiebert and Jayasimha only provide quantitative results for a hypercube configuration. Is their technique best demonstrated on a hypercube? How would it perform on a torus or a mesh? More configurations would have been helpful in assessing the relative performance of their algorithms.

A more serious shortcoming of the work is their choice of metric for evaluation, the "degree of adaptiveness". Though this metric is well-suited for evaluation of a theoretical model, it provides no answers to the questions needed by actual implementors: how do their algorithms perform on actual traffic simulations? How does the increase in degree of adaptiveness translate to real performance gain? Comparing their implementation with Duato's or to the e-cube algorithm for various traffic patterns (random, shuffle, bit-reversed, and so on) would have provided greater insight into their algorithmic improvements.

Pinkston and Warnakulasuriya, in contrast, provide a multitude of experimental results to support their theory, and their methodology seems sound (although it would be interesting to see a comparison between Disha and newer algorithms such as Schwiebert and Jayasimha's instead of to Duato's older algorithm). However, though Disha likely has a protocol for recovering from deadlock, this protocol does not seem to figure into their comparisons at all.

An argument above notes that recovery from deadlock likely requires another set of channels on which the recovery will be made. This family of escape channels is almost certainly a more lightweight channel than a normal family of virtual channels; a single channel per node would suffice, for instance. However, the cost may not be in the interconnection fabric but instead in the routing internals at each node; both the switch and the allocators must allow for routing on normal channels as well as the escape channel. While the internals of routing to the escape channel may not necessarily be as complex or high-performance as to other channels, the connection still must be made at some cost in hardware complexity. Thus, the costs of the recovery hardware are not insignificant and a fair comparison of a deadlock recovery scheme against a deadlock avoidance scheme must include these costs.

Pinkston and Warnakulasuriya also do no comparison of the costs of deadlock recovery against the costs of normal routing. If recovery from deadlock is a lengthy and difficult process that brings the entire fabric to its knees, this cost is significant and should also be added to their comparisons.

Conclusions Despite these shortcomings, both papers build on the work of Duato and that of previous authors, demonstrating clear advantages over their previous work.

With technology trends leading toward more complicated switching fabrics with more flexible routing algorithms as well as more channels with which to route, deadlock recovery looks more and more attractive. However, more data must be acquired on the costs of the method of recovery; if it can be made small, the efficiencies gained from the additional routing freedom should be significant.

An interesting extension to the work of both sets of authors would be the consideration of fault tolerance and its effects on their algorithms. Could the enhanced routing algorithm be extended at low cost to handle the loss of a link or a node? How would it compare to other deadlock avoidance schemes? What complexity would need to be added to a deadlock recovery implementation's escape fabric given the possible loss of a link or a node? Does the increased routing freedom of a Disha-type architecture afford an advantage over competing deadlock avoidance architectures?

It may be possible or even necessary to implement a combination of the two schemes in the presence of faults, beginning with one scheme and falling back to the other if faults were to occur; a flexible, programmable switching architecture would seem to be an ideal testbed for exploring this issue.

References

[1] K. V. Anjan and Timothy M. Pinkston. An efficient, fully adaptive deadlock recovery scheme: Disha. In *Proceedings aof the 22nd International Symposium on Computer Architecture*, pages 201–210, June 1995.

[2] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, C-36(5):547–553, May 1987.

[3] José Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, 1993.

[4] José Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection Networks: An Engineering Approach.* IEEE Computer Society, Los Alamitos, California, 1997.

[5] C. Glass and L. M. Ni. The turn model for adaptive routing. *Journal of the Association for Computing Machinery*, 41(5):874–902, September 1994.

[6] Timothy Mark Pinkston and Sugath Warnakulasuriya. On deadlocks in interconnection networks. *Proceedings of the 24th International Symposium on Computer Achitecture*, pages 38–49, June 1997.

[7] Loren Schwiebert and D. N. Jayasimha. A universal proof technique for deadlock-free routing in interconnection networks. *Papers of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 175–184, 1995.

[8] Sugath Warnakulasuriya and Timothy M. Pinkston. Characterization of deadlocks in interconnection networks. In *Proceedings of the 11th International Parallel Processing Symposium*, Geneva, Switzerland, April 1997.

[9] Sugath Warnakulasuriya and Timothy M. Pinkston. Implementation of deadlock detection in a simulated interconnection network environment. Technical Report CENG 97-01, University of Southern California, January 1997.