Non-Uniformities Introduced by Virtual Channel Deadlock Prevention

Kevin Bolding

July 21, 1992

Abstract

A common scheme for preventing deadlock in networks is the virtual channel method of Dally and Seitz [DS87]. Due to the nature of this scheme, an otherwise completely uniform network will have non-uniformities introduced into it. The variations introduce several effects, ranging from limitations on overall network performance to differences in observed network characteristics from node to node and from message to message.

1 Introduction

A useful multicomputer network must be both efficient and reliable. A key component of reliability is freedom from deadlock. Many schemes have been introduced which prevent deadlock in general networks, at varying costs in terms of additional buffers and complexity. A particularly interesting scheme is presented by Dally and Seitz [DS87] which relies on *virtual channels* to break dependency cycles in a network. The general scheme is to note where cycles may exist in a network and routing scheme and then to add virtual channels in order to transform the circular dependencies into spirals which are, by nature, acyclic.

While *open-ended* k-ary d-cubes (*e.g.* meshes and hypercubes) can be made deadlock-free by restricting routing to be oblivious and dimension ordered, this cannot be extended to k-ary d-cubes where the edges are "wrapped around" (*e.g.* tori). A scheme presented in [DS87] achieves deadlock freedom in wrapped-around networks by dividing each physical channel into two virtual channels and restricting the routing to be dimension ordered. A further restriction prevents deadlock: a message uses the "high" set of virtual channels if the number of the destination address is higher than the current node, and the "low" set of virtual channels if the destination address is less than or equal to the current node address. This prevents deadlock by choosing a *terminal* node at which dependencies in a particular set of virtual channels must terminate, thus preventing cyclic dependencies from occurring.

To achieve maximum performance in a particular network, each of the resources in the network should be loaded equally. If one component reaches saturation before the rest, the network will tend to slow to the load which can be handled by the saturated component.[1] In a sense, this node has become the "weakest link" in a chain, and the strength of the entire chain is thus diminished. This, for example, is responsible for early saturation in mesh networks where there is extra congestion in the center of the network. A torus network exhibits *vertex transitivity* which means that there are no observable differences between any two nodes in the network. Thus, torus networks should not have the problem of congestion in the center because there is

[1] This is true for uniform random traffic, but with non-uniform traffic local saturations may not spread to cover the entire network.

no distinction between nodes in the center and nodes along the edges of a torus. However, when the virtual channel deadlock scheme is implemented, the routing mechanism is perturbed in a manner which re-introduces non-uniformities by restricting which messages may use particular buffers.

2 Uneven buffer utilization

Although the virtual channel scheme successfully prevents deadlock, it does so by the addition of underutilized buffers. Furthermore, the effective buffer space available to each node varies according to the node's position in the network, creating non-uniformities in the network.

Consider the set P of all paths between any two nodes allowed by the network and routing algorithm. Each channel c in the network is part of a set of paths p_c which use c. Also, each virtual channel c_v of channel c has a set of paths p_{c_v} which use c_v. We know that $|p_{c_v}| \leq |p_c|$ since no virtual channel can have a greater number of paths through it than the channel to which it is attached. Now define the virtual channel utilization $u_{c_v} = \frac{|p_{c_v}|}{|p_c|}$ for each virtual channel. ($\sum_{c_v} u_{c_v} = 1$) Thus, u_{c_v} is the fraction of the paths through channel c which use virtual channel c_v.

Now, since each channel is represented by multiple virtual channels, the most effective use of the buffer space provided by the virtual channels is for each virtual channel to have the same fraction of the total traffic. If one virtual channel has more traffic than the others, it becomes the bottleneck for the channel. Thus we define the effective buffer size $B = \frac{1}{\max_{c_v}(u_{c_v})}$. If all the buffers are utilized equally, B will equal the number of buffers; if only one buffer is utilized, B will equal 1.

2.1 Unidirectional channels

Consider a unidirectional ring network under random traffic. This can be thought of as a "slice" of a torus, and since, in dimension-order routing, interactions between dimensions are minimal, the analysis extends to multi-dimensional tori as well. For channel c, $|p_c| = \sum_{i=0}^{N-1} i$.

The scheme proposed in [DS87] routes messages on the high virtual channels if the node number of the source node is less than the destination node, and the low virtual channels otherwise. This results in the "terminal" node being node zero. The number of paths through the high virtual channels is $|p_{c_H}| = \sum_{i=0}^{j} i$ for node j. The number of paths through the low virtual channels is $|p_{c_L}| = \sum_{i=j+1}^{N-1} i$. Thus, for a unidirectional ring, the effective buffer size is:

$$B(j) = \frac{\sum_{i=0}^{N-1} i}{\max\left(\sum_{i=0}^{j} i, \sum_{i=j+1}^{N-1} i\right)}$$

The scheme implemented in designing the torus routing chip [DS86] injects all messages (except from node zero) on the low virtual channels. Messages are routed on the low channels until they cross node zero, when they switch to the high virtual channels and continue on them until they reach their destination. This also results in the terminal node being node zero. The number of paths through the high virtual channels is $|p_{c_H}| = \sum_{i=N-j}^{N-1} i$ for node j. The number of paths through the low virtual channels is $|p_{c_L}| = \sum_{i=0}^{N-j} i$. Thus, for a unidirectional ring,

Node	Dest < Current			Cross 0		
	$\lvert p_{c_H} \rvert$	$\lvert p_{c_L} \rvert$	B	$\lvert p_{c_H} \rvert$	$\lvert p_{c_L} \rvert$	B
0	0	120	1.000	0	120	1.000
1	1	119	1.008	15	105	1.143
2	3	117	1.026	29	91	1.319
3	6	114	1.053	42	78	1.538
4	10	110	1.091	54	66	1.818
5	15	105	1.143	65	55	1.846
6	21	99	1.212	75	45	1.600
7	28	92	1.304	84	36	1.429
8	36	84	1.429	92	28	1.304
9	45	75	1.600	99	21	1.212
10	55	65	1.846	105	15	1.143
11	66	54	1.818	110	10	1.091
12	78	42	1.538	114	6	1.053
13	91	29	1.319	117	3	1.026
14	105	15	1.143	119	1	1.008
15	120	0	1.000	120	0	1.000

Table 1: Effective buffer size for a 16-node unidirectional ring.

the effective buffer size is:

$$B(j) = \frac{\sum_{i=0}^{N-1} i}{\max\left(\sum_{i=N-j}^{N-1} i, \sum_{i=0}^{N-j} i\right)}$$

The number of paths through each set of virtual channels, as well as effective buffer size, are shown in Table 1.

2.2 Bidirectional channels

For a bidirectional ring, the number of paths through any channel is different because of the shorter path lengths. For channel c, $|p_c| = \sum_{i=0}^{N/2} i$.

For the "dest < current" scheme, the number of paths through the high virtual channels is $|p_{c_H}| = \sum_{i=0}^{j-(N-1)/2} i$ for node j. The number of paths through the low virtual channels is $|p_{c_L}| = \sum_{i=j-(N-1)/2+1}^{N/2} i$.[2] Thus, for a bidirectional ring, the effective buffer size is:

$$B(j) = \frac{\sum_{i=0}^{\frac{N}{2}} i}{\max\left(\sum_{i=0}^{j-\frac{N-1}{2}} i, \sum_{i=j-\frac{N-1}{2}+1}^{\frac{N}{2}} i\right)}$$

[2] We define the sum where the lower bound is negative to be the sum from a lower bound of zero.

Node	Dest < Current			Cross 0		
	$\lvert p_{c_H} \rvert$	$\lvert p_{c_L} \rvert$	B	$\lvert p_{c_H} \rvert$	$\lvert p_{c_L} \rvert$	B
0	0	36	1.000	0	36	1.000
1	0	36	1.000	8	28	1.286
2	0	36	1.000	15	21	1.714
3	0	36	1.000	21	15	1.714
4	0	36	1.000	26	10	1.385
5	0	36	1.000	30	6	1.200
6	0	36	1.000	33	3	1.091
7	0	36	1.000	35	1	1.029
8	1	35	1.029	36	0	1.000
9	3	33	1.091	36	0	1.000
10	6	30	1.200	36	0	1.000
11	10	26	1.385	36	0	1.000
12	15	21	1.714	36	0	1.000
13	21	15	1.714	36	0	1.000
14	28	8	1.286	36	0	1.000
15	36	0	1.000	36	0	1.000

Table 2: Effective buffer size for a 16-node bidirectional ring.

For the "crossing node zero" scheme, the number of paths through the high virtual channels is $|p_{c_H}| = \sum_{i=N/2-j+1}^{N/2} i$ for node j. The number of paths through the low virtual channels is $|p_{c_L}| = \sum_{i=0}^{N/2-j} i$. Thus, for a bidirectional ring, the effective buffer size is:

$$B(j) = \frac{\sum_{i=0}^{\frac{N}{2}} i}{\max\left(\sum_{i=\frac{N}{2}-j+1}^{\frac{N}{2}} i, \sum_{i=0}^{\frac{N}{2}-j} i\right)}$$

The number of paths through each set of virtual channels, as well as effective buffer size, are shown in Table 2.

3 Significance

The effect of uneven buffer utilization on performance varies with the amount of buffering in the network. If wormhole routing is implemented with minimal (i.e., single flit) buffering per virtual channel, then performance will not be effected tremendously because the buffers are not playing a critical role in achieving performance. On the other hand, increasing buffering in a network can dramatically boost its performance. Thus, it makes sense to use all of the buffer space available. We see that, in a unidirectional network, the nodes on either side of the "terminal" node have only one usable buffer. In a bidirectional network, this is true for fully one-half of the nodes. Since these nodes become "weakest link" nodes, the network will perform roughly the same as a network which has only one buffer per channel, despite the presence of twice as much buffering capability.

Moreover, the amount of effective buffer space at each node varies from node to node. This results in a very uneven distribution of usable resources within a network and has several negative effects on performance. First of all, the latency which a message experiences depends on which portion of the network it travels through, even under a uniform random load. Ideally, a network will provide uniform performance under a uniform load so that users will not have to worry about being dealt "bad" nodes. Secondly, nodes which have more effective buffer space will have a greater number of opportunities to inject messages into the network, potentially preventing other nodes from injecting messages and resulting in starvation for those nodes. These effects combined to reduce the expected performance of the oblivious torus router described in [BS92].

A third non-uniformity exists which may present different views of network contention to messages which have different distances between their sources and destinations. For the "Dest $<$ Current" scheme, the virtual channel a message uses depends on whether or not the message will *eventually* cross the terminal node (node zero). Consider two messages from the same source which are traveling the same direction at some node common to both of their paths from their source to their destinations. If one of the messages has a destination which is numbered lower than the current node, while the other has a destination which is numbered higher than the current node, the two messages will use different virtual channels. Since they use different virtual channels, they will compete with different messages for buffer space and thus face different levels of congestion. However, from a local point of view, the two messages have arrived on the same channel and wish to leave on the same channel, and should not be distinguishable. Thus, the deadlock prevention mechanism discriminates between messages which should be treated equivalently in a uniform routing strategy. In particular, messages with longer paths from source to destination are more likely to cross the terminal node and be routed on the high virtual channels than messages which have short paths. Because of this, the contention a message faces when competing for buffers differs depending on how far the message must travel in the network.

The same phenomenon exists for the "Crossing node zero" scheme, although the criterion for distinguishing between messages is slightly different. In this case, consider two messages which have the same destination and are at a node common to both of their paths from their sources to their common destination. Again, locally the messages are indistinguishable, as they should, from this point on, follow the same path to their destinations. However, if one of the messages has crossed the zero node previously and the other has not, the two messages will use different virtual channels and, again, face different levels of congestion. Once again, the congestion seen by a particular message at a particular node in the network varies depending on how far apart the message's source and destination are.

An especially disturbing result of this phenomenon is seen with the "Dest $<$ Current" scheme. Two messages being injected into the network from the same node in the same direction, but with different destinations may start out on different virtual channels and face different amounts of congestion during injection. Because of this, it may be easier at some nodes to inject messages which have either short distances to their destinations of long distances to their destinations, depending on the node's location. When some nodes have an easier time injecting their messages into the network than other, starvation becomes a problem. Therefore, with this scheme, a message's likelihood to be starved varies not only with its location in the network, but with the distances that messages it injects into the network need to travel. Fortunately, this problem does not occur in the "Crossing node zero" scheme because all messages at any given node are injected into the same virtual channel.

4 Conclusions

Although the [DS87] scheme provides a simple and efficient method of preventing deadlock in torus networks, it unfortunately introduces non-uniformities in otherwise perfectly uniform networks. The effect of these non-uniformities is quite severe, ranging from differing observed network performance from node to node to a general slowdown of the entire network. Since buffering is critical to achieve good performance in any network, it is not possible to eliminate or minimize the buffering, and thus the problem.

One solution is to associate the buffer space in a node with either the entire node (a shared buffer pool) or with each *physical* channel. Thus, all messages would contend for the same resources, no matter what virtual channel they were routed on. However, in either case, implementation becomes considerably more complex because of the need to create two or more logically separate buffers within one physical buffer.

Because this increases the complexity of the router significantly, it becomes worthwhile to consider abandoning oblivious routing completely and rely on an adaptive technique to prevent deadlock. Non-minimal adaptive routers such as those presented in [NS89, BS92, KS90] prevent deadlock while preserving the node-transitivity of torus networks by relying on local rather than global schemes for preventing deadlock. At the same time, adaptive routing networks allow more flexibility in path selection, making these networks better performers in the presence of uneven network utilization.

References

[BS92] Kevin Bolding and Lawrence Snyder. Mesh and torus chaotic routing. In *Advanced Research in VLSI and Parallel Systems: Proceedings of the 1992 Brown/MIT Conference*, pages 333–347, March 1992.

[DS86] W. Dally and C. Seitz. The torus routing chip. *Journal of Distributed Computing*, 1(3), 1986.

[DS87] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[KS90] Smaragda Konstantinidou and Lawrence Snyder. The chaos router: A practical application of randomization in network routing. In *Proceedings of the 2nd Symposium on Parallel Algorithms and Architectures*, pages 21–30. ACM, 1990.

[NS89] J. Y. Ngai and C. L. Seitz. A framework for adaptive routing in multicomputer networks. In *Proceedings of the Symposium of Parallel Algorithms and Architectures*, pages 1–9. ACM, 1989.