PERFORMANCE OF PIPELINED K-ARY N-CUBE NETWORKS

by

Steven L. Scott and James R. Goodman

Computer Sciences Technical Report #1010

February 1991

Performance of Pipelined *k*-ary *n*-cube Networks

Steven L. Scott and James R. Goodman

Computer Sciences Department
University of Wisconsin - Madison
1210 W. Dayton Street
Madison, WI 53706

Abstract

In a *pipelined* communication network, multiple bits may be simultaneously in flight on a single wire. This allows the cycle time of the network to be independent of the wire lengths, significantly affecting the network design tradeoffs. This paper investigates the design and performance of pipelined *k*-ary *n*-cube networks, with particular emphasis on the choice of dimensionality and radix. Pipelined networks are shown to provide lower latency and higher bandwidth than their synchronous counterparts, especially for high dimensional networks. The optimal dimensionality of pipelined networks is found to be higher than that of synchronous networks. Pipelined networks should be grown by increasing the dimensionality, leaving the radix roughly constant. The effects of switching overhead, message lengths and queue sizes are also investigated. We indicate where results for pipelined networks agree with and differ from previous results obtained for synchronous networks. Our analysis and simulations take switching, decoding, and transmission delays into account. Networks are investigated under the constant link width, constant node size and constant bisection width constraints.

Steven Scott was supported by a fellowship from the Fannie and John Hertz Foundation. James Goodman was supported in part by National Science Foundation grant #CCR-892766.

1. INTRODUCTION

The interconnection network used in a multiprocessor or multicomputer can have a major impact on system performance and cost. Understanding the tradeoffs in network design thus becomes a critical requirement for building cost-effective, high-performance systems. A wide variety of interconnection networks have been proposed (see [Feng81] or [Sieg79] for a summary), each of which can be classified as either *direct* or *indirect*. Indirect networks, such as the omega network [Lawr75], connect processors and memories through multiple intermediate stages of switching elements. The performance of these networks has been extensively analyzed in the literature [Krus83, Pate81, Dias83, Lee84, Yoon87]. Direct networks incorporate the processing elements within the network itself, allowing for direct communication between processors [Seit84]. The performance of these networks has been the subject of more recent attention [Abra89, Agar91, Dall90, Adve91]. Direct networks are gaining in popularity and have been employed in many recent existing or proposed machines, including the Connection Machine [Hill85], Intel iPSC, Cosmic Cube [Seit85], Tera supercomputer [Alve90], CMU-Intel iWarp [Bork88], and Stanford DASH multiprocessor [Leno89].

The most commonly used direct networks are variants of the k-ary n-cube [Sull77]. The k-ary n-cube consists of $N=k^n$ nodes, arranged in n dimensions, with k nodes per dimension (n is the dimensionality, k is the radix). Figure 1 illustrates a 4-ary 3-cube. A node is connected via a direct link to its nearest neighbors in each of n dimensions. Links can be bi- or uni-directional, and, if bi-directional, the wrap-around links may be omitted. Examples of k-ary n-cubes include the ring (n=1), 2D mesh or torus (n=2), 3D torus (n=3) and hypercube (k=2). The generality and flexibility of the k-ary n-cube make it an excellent choice for investigating network design tradeoffs.

For a given system size, the primary design choice is the dimensionality of the network. The default assumption, when varying the dimensionality of a network, is to keep the width of the network links fixed. Design constraints, however, may limit the freedom to do this. Dally has investigated network performance under the *constant bisection width* constraint [Dall90]. This constraint is motivated by wiring area limitations in VLSI, and holds the number of wires crossing the bisection of a network constant as the dimensionality is varied. This causes the link width to decrease as dimensionality is increased. Agarwal has investigated network performance under the *constant node size* constraint as well as the default *constant link width* constraint [Agar91]. The constant node size constraint is motivated by pin limitations on boards and chips, and holds the number of wires per network node constant as the dimensionality is varied. This also causes the link width to decrease as dimensionality is increased, but to a lesser extent than the constant bisection width constraint.

The dimensionality of a network can affect wire lengths, as well as link widths. When the number of logical dimensions exceeds the number of physical dimensions in which the network is constructed, nodes can no longer be connected solely to their physical neighbors, and the length of the longest wire must grow. This will be discussed further in Section 2.2.1. Dally's model of communication latency included wire delay, but did not include delay through the network switches. Coupled with the constant bisection constraint, this led to the conclusion

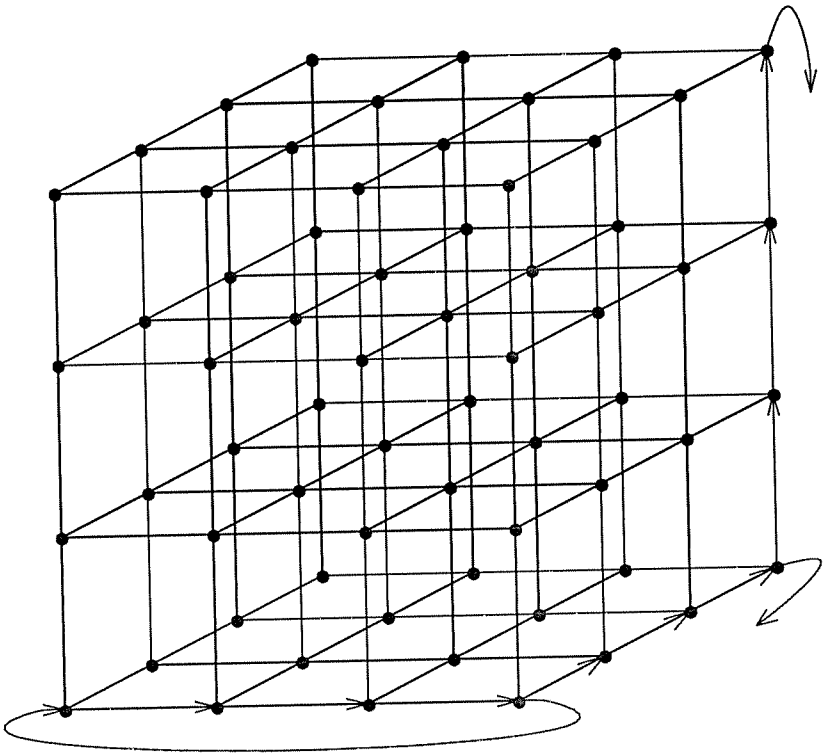


Figure 1: A 4-ary 3-cube Network (N=64, k=4, n=3)

that low dimensional networks (2 to 3 dimensions) provided superior performance for up to many thousands of nodes. Agarwal's model included switching overhead. Coupled with the looser constraints, this argued for higher dimensionality than did Dally's model.

In this paper, we investigate the effect of *pipelining* the transmission of data along the network wires. By allowing multiple bits to be in flight on the same wire, the throughput along a link is divorced from the latency. This has a significant impact on the network design tradeoffs. Previous work has assumed *synchronous* networks, in which the cycle time of the network must include the transmission time across the longest wire in the network. This exacts a heavy penalty on high-dimensional networks because their longer wires give rise to slower cycle times. Pipelined networks are *asynchronous*, meaning that data is clocked onto the wires at a rate determined by switching speed and independent of wire length. This type of network is used in the IEEE Scalable Coherent Interface (SCI) standard [IEEE90].

The remainder of this paper presents a performance study of pipelined k-ary n-cube networks, with particular emphasis on how the design tradeoffs differ from those of synchronous networks. Section 2 develops equations for latency in an unloaded network. We discuss the implementation of a pipelined network, and show how pipelining changes the network design tradeoffs, arguing for higher dimensionality than with synchronous networks. Section 3 discusses network bandwidth and presents simulation results for synchronous and pipelined

networks. Section 4 investigates the effects of switching overhead, message length, and queue sizes. Finally, concluding remarks are presented in Section 5.

2. UNLOADED LATENCY

An important metric of network performance is the mean latency of message transmission in the absence of contention (the unloaded latency). Both switching and wire transmission delay contribute to this latency. The manner in which switching and transmission delays interact is a primary difference between pipelined and synchronous networks. This will be further explained in the following sections. In Section 2.1 we describe our network node model and other assumptions, and in Section 2.2 we derive formulas for the unloaded latency in pipelined and synchronous networks.

2.1. Model and Assumptions

Recall that a k-ary n-cube network consists of $N=k^n$ nodes. Our model of a network node appears in Figure 2. Routing a message between two points consists of transmitting the message around n rings, one in each dimension. A dimension is skipped if the message source and destination have the same coordinate in that dimension. A low-level protocol handles the transmission around each ring, independent of the other dimensions and any higher-level, end-to-end protocol. The ring interfaces are based loosely on the SCI logical layer protocol [IEEE90]. The links are unidirectional and W bits wide. A message of L bits is decomposed into P *flits*, with P given by

$$P = \left\lceil \frac{L}{W} \right\rceil \tag{1}$$

Routing is similar to *virtual cut through* routing, discussed in [Kerm79]. Upon being placed in the input queue by the CPU, a message is switched to the output queue for the appropriate dimension and gated onto the output link (T_{switch} cycles). It then uses T_{wire} cycles to travel to the next network node. When the head of the message arrives at a node, T_{decode} cycles are spent decoding the message. The number of decode cycles is determined by the number of flits needed to form the node address of the message destination:

$$T_{decode} = \left\lceil \frac{\log_2 N}{W} \right\rceil \tag{2}$$

If continuing in the current dimension, the message is routed through the ring buffer and gated onto the ring output link (T_{pass} cycles). If changing dimensions, the message is routed through the input queue, switched to the appropriate output queue and gated onto the output link for the new dimension (T_{switch} cycles).

We will assume values of T_{pass}=1 and T_{switch}=2. The overhead for changing dimensions is greater than that for continuing in the same dimension, due to the extra switching step involved. To reduce the complexity and increase switching speed, the switch could be implemented as a n-element ring. Contention on the ring would be

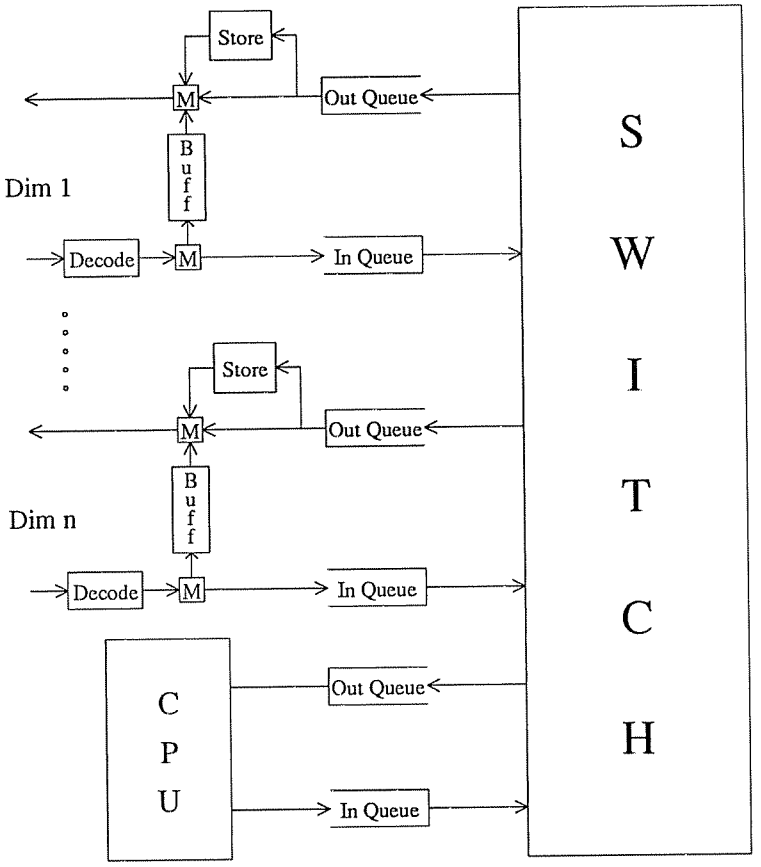


Figure 2: A Node from a Pipelined k-ary n-cube Network

minimal, because the majority of the traffic would be routing to the next dimension and would thus need to travel only one hop along the ring. In this case, of course, the switching delay would be greater for messages changing to other than the next highest dimension.

Low-level routing in each dimension is based on the logical layer of the IEEE SCI protocol [IEEE90]. When a message is removed from a ring (either switched to the processor or to a different dimension), an *echo packet* is routed the remainder of the way around the ring to acknowledge the receipt of the message on this ring. Messages are stored at the node in which they first enter a ring until the ring acknowledgement is received. It is possible that an input queue will have insufficient space to accept a message (due to contention), in which case a *negative acknowledgement* is returned around the ring instead. When a negative acknowledgement is received, the message is retransmitted. Ring acknowledgements are needed due to the asynchronous, pipelined nature of the networks. Acknowledgements are provided on a per-message basis after a complete ring traversal, rather than a per-flit basis on each link, as is done in a synchronous network. Acknowledgements can also provide fault tolerance by checking a CRC at the end of the message [IEEE90]. Unlike the input queue, the ring buffer is

guaranteed to be able to accept a flit on every cycle. The output queue is given priority to initiate a message transmission on an output link, but only if the ring buffer has enough free space to hold a message of equal length. In this way the ring buffer cannot fill up before the message has been completely drained from the output queue.

2.2. Latency Equations

The unloaded latency can be derived by simply accounting for the latencies described in Section 2.1. The mean number of hops traveled by a message is given by

$$mean\ hops = n\left[\frac{k-1}{2}\right] \tag{3}$$

Since our switching overhead is greater for changing dimensions than for continuing in the same dimension, we must break this down by dimension. A message routes in a dimension with probability $\frac{k-1}{k}$, and traverses a mean of $\frac{k}{2}$ hops in a taken dimension. The total mean unloaded latency, in cycles, is

$$Latency_{pipe} = T_{switch} + n\left[\frac{k-1}{k}\right]\left[\frac{k}{2}(T_{wire}+T_{decode}) + \left[\frac{k-2}{2}\right]T_{pass}+T_{switch}\right] + P - 1 \tag{4}$$

Note that the delay due to wire transmission is *added* to decoding and switching delays; it does *not* affect the switch cycle time. This allows the cycle time to be kept small and prevents transmission delay from affecting the queueing delay in the switches and the delay between the head and tail of a message. The value of T_{wire} and W will be explored in the following sections.

2.2.1. Wire length

The wire length is not necessarily equal for all links. We assume that the network is being implemented in 3 physical dimensions. This differs from [Dall90], where only 2 dimensions are assumed (primarily to reflect the two-dimensional nature of an all-VLSI implementation). For networks with 3 or fewer logical dimensions, wire lengths are independent of system size and uniformly short.[1] Let S be the ratio of switch cycle time to wire transmission delay for a 3-dimensional network. Since the wire delay must round up to an integral number of cycles, T_{wire} is equal to $\lceil 1/S \rceil$. For a network with $n \geq 3$ dimensions, we can no longer run all links between physical neighbors. We embed $n/3$ logical dimensions in each physical dimension, and the length of the *longest* wire

[1] In order to avoid the long wrap-around wire illustrated in Figure 1, an actual implementation can weave the links of a ring such that half of the nodes in the ring are visited left to right, and the other half visited right to left. This is referred to as a folded torus in [Dall90]. We note that this introduces a factor of 2 increase in wire lengths and hereafter deal with non-folded networks and ignore the wraparound overhead for simplicity without loss of generality.

in the system grows as $\left\lceil k^{n/3-1} \right\rceil$. The maximum wire delay is then given by

$$T_{wire_{\max}} = \left\lceil \frac{k^{\frac{n}{3}-1}}{S} \right\rceil \tag{5}$$

In synchronous networks, the cycle time must accommodate the transmission delay over the longest wire, and thus the maximum wire delay is effectively suffered over *all* links. This has the effect of increasing the total delay due to wire transmission as the dimensionality of a network is increased, even though the mean number of *hops* between two nodes decreases. This has been a severe handicap to higher-dimensional networks in previous studies [Dall90, Agar91]. In a pipelined network, however, the number of cycles spent traversing a given wire is a function of that wire's length only; increased wire delay is suffered only for the longer wires. The mean wire delay, therefore, is determined by the *mean* wire length. The mean wire length grows when the number of logical dimensions exceeds the number of physical dimensions, but the *total* wire traversed between nodes does *not* grow. Figure 3 illustrates this point by showing 4 logical dimensions embedded in a single physical dimension. The node on the left can reach any of the other 15 nodes by following up to 4 of the links shown with arrows. The mean number of links traversed is less than it would be with only nearest-neighbor links, but the mean link length is longer. The two factors cancel out, and the mean distance traveled is the same in either case; the higher-dimensional network does not cause backtracking.[2]

We can calculate the mean wire length in an n-dimensional network by dividing the mean distance between nodes (number of hops in a 3-dimensional network) by the mean number of hops in the n-dimensional network.

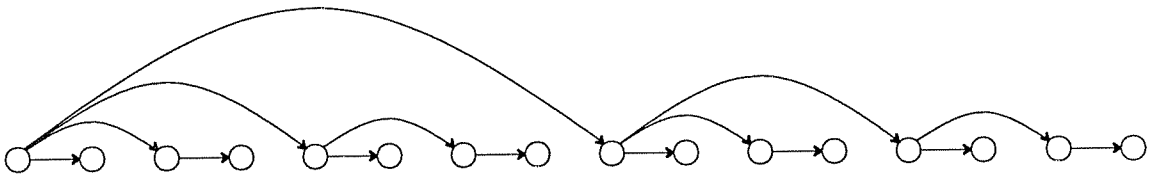


Figure 3: Four logical dimensions embedded in one physical dimension

[2]If the number of logical dimensions is not an even multiple of the number of physical dimensions, then there will actually be a *small* amount of backtracking in the remaining logical dimensions, causing the mean distance traveled to be slightly higher for high-dimensional networks. However, high-dimensional networks also save some wire length by reducing the wrap-around penalty and reducing the number of off-board or off-chip transitions. Both these effects will be ignored.

The mean distance is $3\left\lceil\frac{k^{1/3}-1}{2}\right\rceil$, and the mean number of hops is given in equation (3). The mean wire length when $n \geq 3$ is thus

$$mean\ wire\ length = \left\lceil\frac{k^{n/3}-1}{k-1}\right\rceil\left(\frac{3}{n}\right) \tag{6}$$

and $T_{wire} \approx \lceil mean\ wire\ length \rceil$. To obtain an accurate value for T_{wire}, the wire transmission delay must be rounded up to a whole number for each wire (we cannot simply round up the mean). There are $n/3$ logical dimensions embedded completely in each physical dimension. The wire length for the j^{th} logical dimension in a physical dimension is $\left\lceil k^{(\frac{n}{3}-j)}\right\rceil$. Another $n\ mod\ 3$ logical dimensions are embedded amongst all 3 physical dimensions. The wire length for the j^{th} "extra" dimension can be set to j. For each wire length l, the transmission delay is $\lceil l/S \rceil$.

They key attribute of pipelined networks with respect to wire length, is that the cycle time is decoupled from the transmission delay, so the maximum wire length does not determine network performance. The total wire traversed by a message is roughly independent of the dimensionality of the network. As such, wire length becomes a non-issue when determining the optimal dimensionality of the network.

2.2.2. Link width

The link width W is affected by the dimensionality of the network and the constraint under which it is being designed. The default assumption is the constant link width constraint. It assumes that the link width, which determines the number of flits into which a message must be decomposed, is independent of the dimensionality of the network. Since the number of wires attached to a node is equal to $2nW$, this causes the total number of wires per node to increase as the dimensionality of a network is increased. The constant node size constraint keeps the wires per node fixed as the dimensionality is varied. The link width is then some constant divided by n. The constant bisection constraint keeps the number of wires across the bisection fixed as the dimensionality is varied. The number of wires across the bisection is $2Wk^{n-1}$, so link width is given by

$$W_{const\ bisec} = k\left\lceil\frac{bisection\ width}{2N}\right\rceil \tag{7}$$

The constant bisection constraint is used in [Dall90] in order to reflect the limited wiring area of a network implemented entirely in VLSI. We believe that for a multiprocessor implemented across several boards, the constant node size constraint is much more realistic. This is due both to pin limitations off-chip and off-board. Depending upon the technology used to implement the network, the constant link width constraint may be realistic as well. Intranode data paths, for example, may dictate the link width and the pin limitations may not be restrictive. In [Agar91], all three constraints are considered, with the emphasis on the constant link width constraint.

2.2.3. Synchronous networks

In a synchronous network, the cycle time must include both switch operation and wire transmission. As such, the maximum wire transmission delay has a multiplicative effect on overall latency. The synchronous latency is given by

$$Latency_{synch} = \left(1+T_{wire_{max}}\right)\left\{T_{switch}+n\left\lceil\frac{k-1}{k}\right\rceil\left[\frac{k}{2}T_{decode}+\left\lceil\frac{k-2}{2}\right\rceil T_{pass}+T_{switch}\right]+P-1\right\} \quad (8)$$

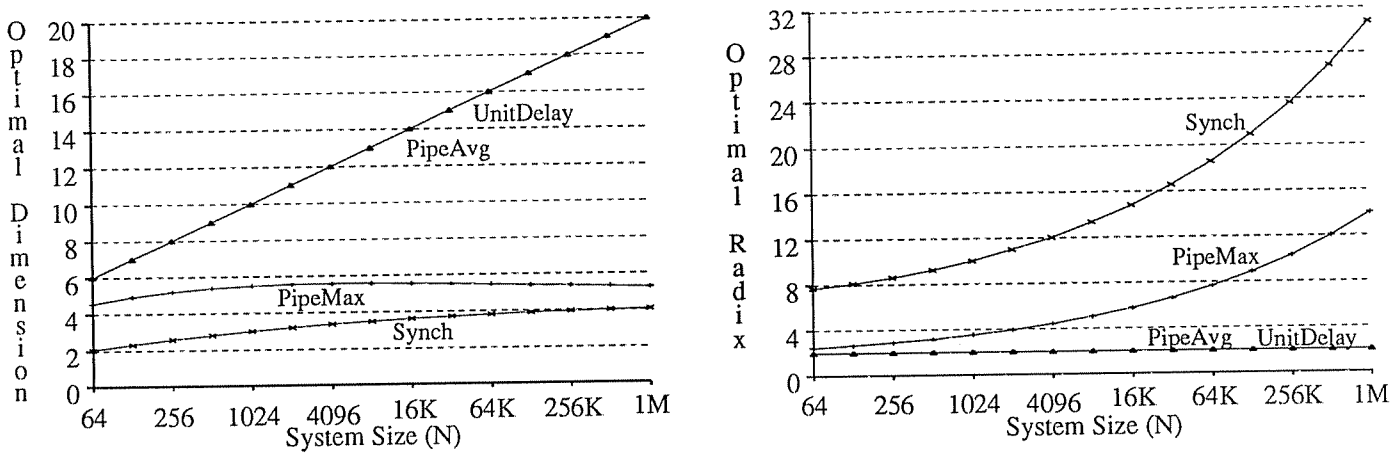
The time unit in equation (8) is the switch cycle time *not* including wire transmission delay, which allows for direct comparison with equation (4). The first factor, $\left(1+T_{wire_{max}}\right)$, represents the increased cycle time of the synchronous network due to wire transmission delay. The remaining factor (enclosed in curly braces) represents the number of cycles used to transmit the message. Equation (8) is the same as equation (4), save that wire delay is taken out of the cycle *count*, and instead is used to increase the cycle *length*.

2.3. Optimal Dimensionality as System Size Grows

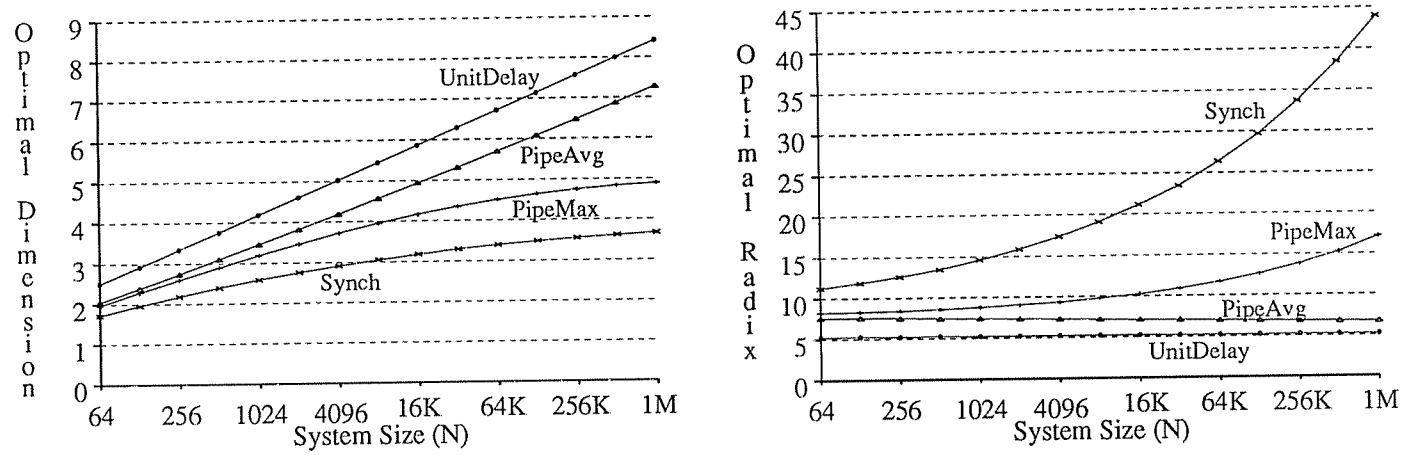
Using equations (4) and (8), we can now calculate the dimensionality that gives the lowest unloaded latency for a given system size and constraint. We will consider several network variants. The first, *Synch*, assumes a synchronous network, and is characterized by equation (8). The second, *PipeAvg*, assumes a pipelined network, and is characterized by equation (4), with T_{wire} derived using the mean wire length. We also consider two unrealistic networks for purposes of comparison. The first, *PipeMax*, assumes a pipelined network, but uses the *maximum* wire length to derive T_{wire}. This network is unnecessarily conservative, but allows us to separately identify the two main performance advantages of pipelined networks. The difference between *Synch* and *PipeMax* is due to the de-coupling of cycle time from transmission latency. The difference between *PipeMax* and *PipeAvg* is due to the fact that the wire delay in a pipelined network is determined by the mean wire length, rather than the maximum wire length. The second unrealistic network, *UnitDelay*, assumes unit wire delays for all links. This has been a common assumption in previous studies.

Figure 4 plots the optimal dimensionality and radix of a network (based solely on unloaded latency) as system size grows. Note that the optimal dimensionality and optimal radix each are related by the equation $N=k^n$. Parts (a), (b) and (c) assume the constant link width, node size, and bisection width constraints, respectively. Each graph includes results for each of the 4 network variants. The ratio of switch cycle time to base wire delay, S, is 2. The latency is the sum of two message latencies: one with a 16 byte packet and one with an 80 byte packet. These correspond to the address and data packet sizes in SCI [IEEE90].

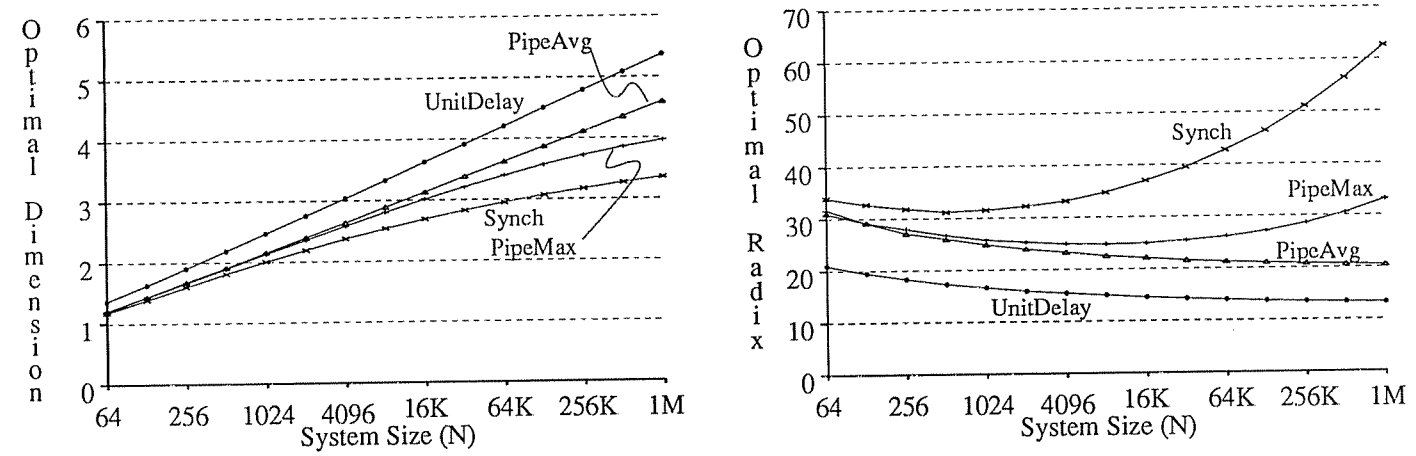
To compute the optimal dimensionality, we have treated all discrete quantities as continuous. Thus, wire and decode delay cycles as well as the dimensionality and radix of the networks can be fractional. When all discrete effects are included, the graphs jump erratically and the general trends cannot be clearly seen. The results are also very sensitive to arbitrary values such as the normalized link width. When we calculate actual latencies



(a) Constant Link Width (W=32)



(b) Constant Node Size ($2nW$=192)



(c) Constant Bisection Width (W=2 for hypercube)

Figure 4: Optimal Dimensionality and Radix versus System Size

in Section 4, we will use only discrete values.

Figure 4(a) assumes a constant link width of 32 bits. Thus the dimensionality does not affect the decode delay or the number of flits. As the dimensionality of a given sized network is increased, the mean number of hops for a message decreases, but wire lengths increase. *UnitDelay* and *PipeAvg* are not affected by wire length increases, so they always perform best with the highest dimensionality (a hypercube) due to the reduction in message hops. *PipeMax* and *Synch* are both affected by wire length, but to varying degrees. The optimal dimensionality for the pipelined network is over 2 greater than that for the synchronous network for system sizes up to 4096. As system size increases to 1M, the wire lengths begin to dominate latency and the optimal dimensionality for *PipeMax* actually decreases slightly. Asymptotically, the optimal dimensionality for both *Synch* and *PipeMax* will go to 3, the number of physical dimensions.

Figure 4(b) assumes a constant node size of 192 wires (W=32 when n=3). The optimal dimensionality for all four networks is lower under this constraint than under the constant link width constraint. This is due to the effect of decreasing the link width as dimensionality is increased. The radix for *UnitDelay* still remains constant as N increases, but the optimal radix is now 5 instead of 2, and the optimal dimensionality is correspondingly lower. There is now a difference between *UnitDelay* and *PipeAvg* due to the fact that the total wire delay for *UnitDelay* decreases when dimensionality is increased. As before, the wire lengths for *PipeMax* and *Synch* become dominant for very large system sizes, causing the optimal radix to increase. The optimal dimensionality for *PipeMax* is still about 1 greater than that for *Synch*.

Figure 4(c) assumes a constant bisection width, normalized to a hypercube with W=2. The optimal dimensionality for all networks is lower than in parts (a) and (b), due to the tighter constraint on link width. For small system sizes, the link width is the dominant effect. As system size increases, the number of hops becomes more important, and the optimal radix decreases. For larger system sizes, the wire length begins to dominate, and the optimal radix for *PipeMax* and *Synch* begins to increase. The ordering of the 4 network types in terms of optimal dimensionality remains the same.

The striking conclusion to be drawn from Figure 4 is that pipelining a network significantly impacts the choice of dimensionality. Pipelining argues for *higher* dimensionality. Let us compare *PipeAvg* to *Synch* (*UnitDelay* is overly optimistic and *PipeMax* is overly pessimistic). The radix of the pipelined network should be kept constant as system size is increased. When link width is unconstrained, the hypercube provides superior performance for all system sizes. When link width is constrained by either the constant node size or bisection width assumptions, the optimal radix increases. Other factors might also affect the optimal radix. An increase in the penalty for switching dimensions (T_{switch}) would increase the optimal radix, and an increase in switch cycle time would decrease the optimal radix. Regardless of the optimal radix, however, the networks should still be grown by increasing the dimensionality, *not* the radix. This is not the case for the synchronous network, where the wire delay directly affects cycle time. The optimal dimensionality is limited by the longer wire lengths for high dimensions, and the optimal radix increases significantly for large system sizes.

The analysis thus far has considered only unloaded latency. When bandwidth is considered, the argument for keeping the radix fixed becomes more compelling. Under uniform traffic, the rate of traffic across a link is proportional to the radix of the network [Scot91]. Thus, if the network is grown by increasing the radix, the rate of traffic per link will also increase. Section 3 explores bandwidth considerations in more detail.

3. THROUGHPUT AND CONTENTION

Real networks carry traffic. To obtain a clear picture of a network's performance, latency must be viewed in conjunction with throughput. Latency rises significantly as the rate of traffic approaches the network's maximum capacity. The maximum capacity is affected by the link width and dimensionality of the network. In this section, the maximum throughput of a network will be derived, and a simulation study of pipelined and synchronous networks will be presented.

The maximum throughput is determined by the message length in bits, L, the number of wires traversed by the message, W_{msg}, and the total number of wires in the network, W_{total}. The maximum throughput, in bits per cycle per processor, is

$$X_{pipe_{\max}} = \frac{L\,W_{total}}{N\,W_{msg}} \tag{9}$$

The number of wires in the network is given by

$$W_{total} = N\,n\,W \tag{10}$$

To derive L and W_{msg}, we assume the use of two message packet formats: address only (L_{addr} bits) and address plus data (L_{data} bits). A fraction f_{data} of the packets generated are data packets, with the remainder being address only. The ring acknowledgement packets described in Section 2.1 are L_{ack} bits. The mean message length in bits is

$$L = f_{data}\,L_{data} + f_{addr}\,L_{addr} \tag{11}$$

When a message traverses a ring, it travels part way as a data or address packet, and the remainder of the way as an acknowledgement packet. Taking this into account, and the fact that the message length must be rounded up to the nearest multiple of W, the total number of wires traversed by a message is given by

$$W_{msg} = W\left\lceil \frac{f_{data}\left\lceil \frac{L_{data}}{W} \right\rceil + f_{addr}\left\lceil \frac{L_{addr}}{W} \right\rceil + \left\lceil \frac{L_{ack}}{W} \right\rceil}{2} n(k-1) \right\rceil \tag{12}$$

Substituting equations (10), (11) and (12) into equation (9) yields

$$X_{pipe_{\max}} = \frac{2(f_{data} L_{data} + f_{addr} L_{addr})}{(k-1)\left(f_{data}\left\lceil \frac{L_{data}}{W} \right\rceil + f_{addr}\left\lceil \frac{L_{addr}}{W} \right\rceil + \left\lceil \frac{L_{ack}}{W} \right\rceil\right)} \simeq \frac{W}{(k-1)} \tag{13}$$

Assuming that packet lengths are an integral number of flits, the maximum throughput is proportional and approximately equal to $\frac{W}{(k-1)}$.

For synchronous networks, the maximum throughput is the same, except that it must be scaled down to reflect the longer cycle time needed to include wire transmission. With the time unit set equal to that in equation (13) (one cycle, not including wire transmission time) to allow direct comparison, the maximum throughput in a synchronous network is

$$X_{synch_{\max}} = \frac{X_{pipe_{\max}}}{1+\frac{k^{n/3-1}}{S}} \tag{14}$$

For synchronous networks, not only does the latency increase as wire lengths grow, but the traffic capacity of the network decreases.

Equations (13) and (14) give throughput maximums, but do not explain the relationship between throughput and latency. For this, either a queueing model or simulation is necessary. The following section describes a simulator that models the network node shown in Figure 2 with great detail and uses an iterative solution technique to efficiently provide performance results for arbitrary network sizes.

3.1. Simulation Model

The simulation model used in this paper is based upon the network described in Section 2.1 and illustrated in Figure 2. Parameters to the model include the radix and dimensionality of the network, the link width, the size of the input queues, output queues and ring buffers, the size of the address and data packets, the ratio of address to data packets and the ratio of switch cycle time to base wire delay. Only a single node is simulated. The arrival of packets from the processor and each ring is a Poisson process with rates determined by the system size. Let the packet arrival rate from the CPU be r, of which a fraction f_{data} are data packets and the remainder address packets. Each packet traverses a mean of $\frac{k-1}{2}$ links in each dimension, and causes an acknowledgment packet to traverse a mean of $\frac{k-1}{2}$ links in each dimension. The base packet arrival rates at each of the n switch input links is thus $(k-1)r$, with $\frac{1}{2}$ the packets being acknowledgements, $\frac{f_{data}}{2}$ of the packets being data packets, and $\frac{1-f_{data}}{2}$ of the packets being address packets. A fraction $\frac{2}{k}$ of all acknowledgement packets received at the switch are destined for this node and can thus be discarded. The remaining acknowledgement packets must be passed along the same dimension. A fraction $\frac{2}{k}$ of all data and address packets are also destined for this node,

with the remainder to be passed along the same dimension. The address and data packets destined for this node are either switched to another dimension or routed to the CPU. A packet arriving from dimension i is switched to dimension $i+j$ with probability $\left[\frac{k-1}{(k-1)^j}\right]$. The packet is switched to the CPU with probability $\left[\frac{1}{k}\right]^{n-i}$.

If all packets were accepted by the switch, then the simulator could use the above arrival rates and routing probabilities to simulate the entire network. The time to pass through the switch in each dimension, and the time to switch into each dimension would be calculated, and these could be used to construct the total latency through the network for the given CPU request rate. However, some packets may not be accepted, causing negative acknowledgements to be returned, which in turn cause the packets to be retransmitted. This not only increases the *latency* through the network (because a packet may have to be sent around a ring more than once), but increases the *traffic* as well.

The relationship between the fraction of packets that are dropped, the base traffic, and the extra traffic generated as a result of dropped packets is as follows (let B be the base traffic, E be the extra traffic, and f be the fraction of dropped packets):

$$(B+E)f = E \quad \Rightarrow \quad E = B\left[\frac{f}{1-f}\right] \quad \Rightarrow \quad (B+E) = \left[\frac{B}{1-f}\right] \tag{15}$$

The simulator uses an iterative technique to arrive at a solution for the above equations. The initial arrival rates are set according to the system size and the processor request rate as described above. The system is simulated and the fraction of dropped packets is calculated for each dimension for both packet types (address and data). These are used to calculate new arrival rates for each dimension and packet type, using the rightmost formula in equation (15). The base rates are calculated used the *realized* throughput of CPU requests from the previous iteration. The arrival rates of acknowledgment packets are simply the base rates, but the arrival rates of data and address packets increase (provided the fraction of dropped packets is nonzero). The arrival rates of negative acknowledgments are equal to the sums of the increased arrival rates for address and data packets.

In order to calculate latency, the following quantities are measured for both address and data packets.

λ_i = fraction of packets dropped in dimension i

α_i = T_{pass} for dimension i

β_i = T_{switch} for packets *leaving* dimension i

γ_i = mean time between accepting a packet and transmitting the corresponding acknowledgement packet

δ_i = mean time between accepting a negative acknowledgement packet and retransmitting the corresponding packet

The corresponding latency is

$$L = P-1+\beta_0+\sum_{i=1}^{n}\left\lceil\frac{k-1}{k}\right\rceil\left[\frac{k}{2}(T_{switch}+T_{decode})+\left\lceil\frac{k-2}{2}\right\rceil\alpha_i+\beta_i+\left\lceil\frac{\lambda_i}{1-\lambda_i}\right\rceil\left[k(T_{switch}+T_{decode})+(k-2)\alpha_i+\gamma_i+\delta_i\right]\right] \quad (16)$$

The throughput, address packet latency and data packet latency are calculated after each iteration, and the iterations continue until the relative change in the results is below 2%. Smaller thresholds are impractical due to the inherent randomness in the simulation output (simulation error becomes dominant within a few iterations). The 90% confidence intervals for the results, derived using the method of batch means, are mostly under 1%, and occasionally as high as 5%.

3.2. Simulation Results

Figures 5 through 10 present system configurations and simulation results for various system sizes, constraints and network types. Parameter values are L_{addr}=128, L_{data}=640, L_{ack}=64, S=2, f_{data}=0.3 and all queue and buffer sizes are 100 bytes.

Figure 5 (a-c) presents alternative configurations for a 4096-node, pipelined network. The dimensionality of the network is varied under the constant link width, node size and bisection width constraints. Each table includes the link width, node size, and bisection width for each configuration, and holds one of the three quantities constant according the the appropriate constraint. Decode delay cycles per hop (determined by the link width) and maximum throughput (in bits per cycle per processor) are also shown. Finally, wire delay cycles per hop and total unloaded network latency are shown under two assumptions: that the maximum wire delay is applied to all wires, and that the actual wire delays are used. The values shown represent actual system configurations; all discrete quantities are indeed discrete. The wire delay when using actual wire lengths is calculated by rounding up wire delays for each dimension before averaging. As in Section 2.2, the latencies are for a round trip message consisting of one address packet followed by one data packet.

Under all constraints, the wire delay increases as the dimensionality is increased. Under the constant node size and bisection width constraints the decode delay increases as well. The maximum throughput increases with dimensionality, but at a different rate, depending upon the constraint. For higher-dimensional networks, the unloaded latencies using maximum wire delays are different than the unloaded latencies using actual wire delays. This can lead to a different choice of optimal dimensionality based upon unloaded latency, as was seen in Figure 4. The latencies under the two assumptions converge under heavy loads, however, because the maximum throughput is the same in both cases.

Figure 6 presents the simulation results for the networks listed in Figure 5. The differing traffic capacities of the networks can be clearly seen. Figure 6(a) shows latency versus throughput for networks under the constant link width constraint. The higher-dimensional networks are clearly superior, providing lower latency for a given throughput and a substantially greater maximum throughput.

Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Decode Delay Cycles	Maximum Throughput	Max Wire Lengths		Actual Wire lengths	
							Wire Delay Cycles	Roundtrip Unloaded Latency	Wire Delay Cycles	Roundtrip Unloaded Latency
2	64	32	128	4096	1	0.837	1	407.9	1.00	407.9
3	16	32	192	16384	1	3.514	1	166.6	1.00	166.6
4	8	32	256	32768	1	7.529	1	117.0	1.00	117.0
6	4	32	384	65536	1	17.569	2	107.0	1.50	98.0
12	2	32	768	131072	1	52.706	4	110.0	2.00	86.0

(a) Constant Link Width

Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Decode Delay Cycles	Maximum Throughput	Max Wire Lengths		Actual Wire lengths	
							Wire Delay Cycles	Roundtrip Unloaded Latency	Wire Delay Cycles	Roundtrip Unloaded Latency
2	64	48	192	6144	1	1.094	1	400.9	1.00	400.9
3	16	32	192	16384	1	3.514	1	166.6	1.00	166.6
4	8	24	192	24576	1	5.333	1	126.0	1.00	126.0
6	4	16	192	32768	1	8.784	2	131.0	1.50	122.0
12	2	8	192	32768	2	13.176	4	194.0	2.00	170.0

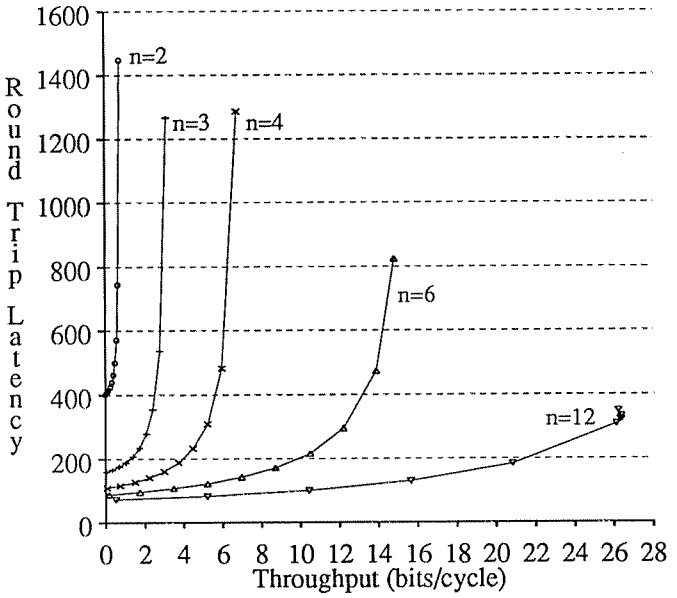
(b) Constant Node Size

Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Decode Delay Cycles	Maximum Throughput	Max Wire Lengths		Actual Wire lengths	
							Wire Delay Cycles	Roundtrip Unloaded Latency	Wire Delay Cycles	Roundtrip Unloaded Latency
2	64	128	512	16384	1	2.844	1	389.9	1.00	389.9
3	16	32	192	16384	1	3.514	1	166.6	1.00	166.6
4	8	16	128	16384	1	3.765	1	141.0	1.00	141.0
6	4	8	96	16384	2	4.392	2	197.0	1.50	188.0
12	2	4	96	16384	3	6.588	4	302.0	2.00	278.0

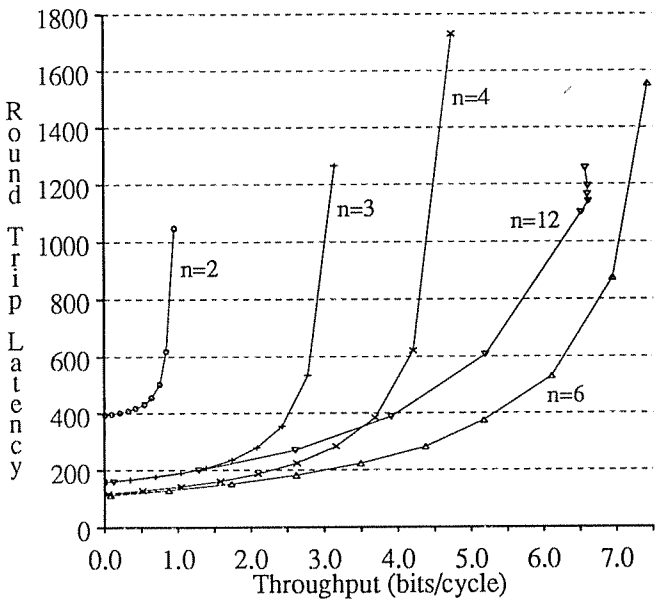
(c) Constant Bisection Width

Figure 5: Actual System Configurations (Pipelined, N=4096)

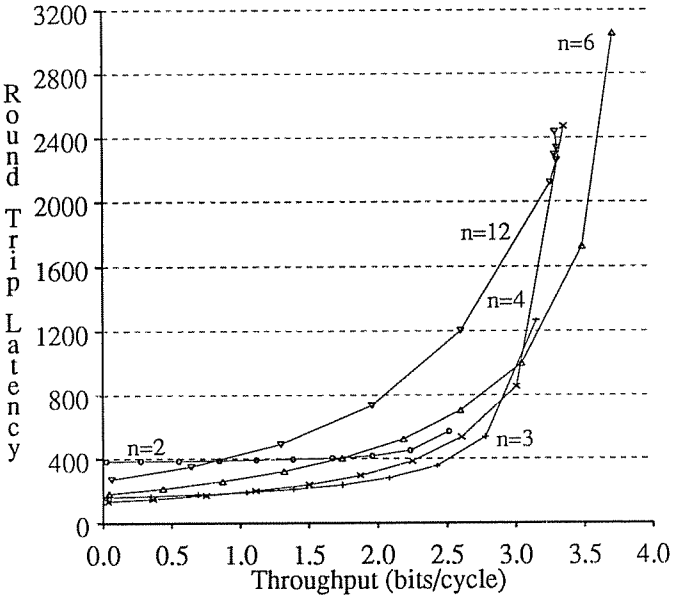
In Figure 6(b), the constant node size constraint is enforced. The 6-dimensional network is superior in this case. Although there is very little difference between the unloaded latency of the 4- and 6-dimensional networks, the 6-dimensional network performs significantly better under heavy traffic. Similarly, the 3- and 12-dimensional networks have nearly equal unloaded latencies (the 3-dimensional latency is slightly lower), but the 12-dimensional network performs better as traffic increases. This illustrates why unloaded latency alone is inadequate to characterize the performance of a network. It is interesting to note that the 6-dimensional network achieves higher throughput than the 12-dimensional network even though it has a lower maximum throughput. This is because of its larger link width, which allows a more uniform utilization of its links.



(a) Constant Link Width



(b) Constant Node Size



(c) Constant Bisection Width

Figure 6: Latency versus Throughput (Pipelined, N=4096)

Figure 6(c) uses the constant bisection width constraint. It is less clear here which network provides the best performance. While the 4-dimensional network has the lowest unloaded latency, the 3-dimensional network

has slightly lower latency under heavier loads. The higher-dimensional networks are severely penalized by smaller link widths and greater decode delays.

Figure 7 presents network configurations for pipelined, 1M-node networks. Several observations can be made in comparing the values to those for 4096-node networks. One is that there is now a significant difference between the unloaded latency when assuming maximum wire delay and the unloaded latency when assuming

Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Decode Delay Cycles	Maximum Throughput	Max Wire Lengths		Actual Wire lengths	
							Wire Delay Cycles	Roundtrip Unloaded Latency	Wire Delay Cycles	Roundtrip Unloaded Latency
2	1024	32	128	65536	1	0.052	1	6168.0	1.00	6168.0
3	102	32	192	665856	1	0.522	1	940.9	1.00	940.9
4	32	32	256	2097152	1	1.700	2	529.7	1.75	498.7
5	16	32	320	4194304	1	3.514	4	485.4	2.80	395.4
10	4	32	640	16777216	1	17.569	13	491.0	5.50	266.0
20	2	32	1280	33554432	1	52.706	26	606.0	8.05	247.0

(a) Constant Link Width

Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Decode Delay Cycles	Maximum Throughput	Max Wire Lengths		Actual Wire lengths	
							Wire Delay Cycles	Roundtrip Unloaded Latency	Wire Delay Cycles	Roundtrip Unloaded Latency
2	1024	40	160	81920	1	0.058	1	6164.0	1.00	6164.0
3	102	26	156	541008	1	0.403	1	946.9	1.00	946.9
4	32	20	160	1310720	1	0.997	2	544.7	1.75	513.7
5	16	16	160	2097152	2	1.757	4	584.4	2.80	494.4
10	4	8	160	4194304	3	4.392	13	623.0	5.50	398.0
20	2	4	160	4194304	5	6.588	26	854.0	8.05	495.0

(b) Constant Node Size

Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Decode Delay Cycles	Maximum Throughput	Max Wire Lengths		Actual Wire lengths	
							Wire Delay Cycles	Roundtrip Unloaded Latency	Wire Delay Cycles	Roundtrip Unloaded Latency
2	1024	512	2048	1048576	1	0.250	1	6147.0	1.00	6147.0
3	102	51	306	1061208	1	0.710	1	932.9	1.00	932.9
4	32	16	128	1048576	2	0.850	2	677.7	1.75	646.7
5	16	8	80	1048576	3	0.878	4	707.4	2.80	617.4
10	4	2	40	1048576	10	1.098	13	1121.0	5.50	896.0
20	2	1	40	1048576	20	1.647	26	1730.0	8.05	1371.0

(c) Constant Bisection Width

Figure 7: Actual System Configurations (Pipelined, N=1048576)

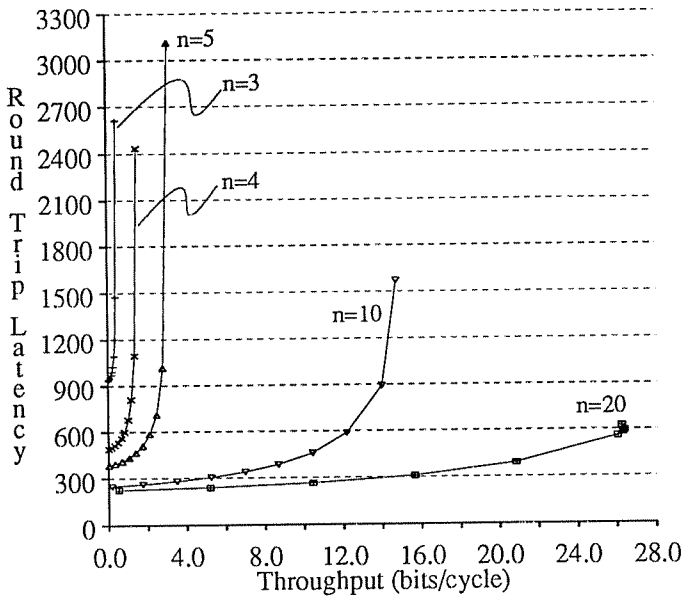
actual wire delays. The *maximum* wire delay is a full 26 cycles for the hypercube (which corresponds to over 50 times the base wire delay), but the *mean* wire delay is only 8.05 cycles. Because pipelined networks do not have to increase their cycle time to accommodate these long transmission times, and because the maximum wire transmission time does not affect the shorter wires, pipelined networks appear especially promising for very large systems. Another interesting observation is that the difference in maximum throughput between the 2- and 20-dimensional networks is *very* large. It is a factor of 1000 under the constant link width constraint. For the constant node size constraint, where the total number of wires in the system is independent of dimensionality, the difference in the maximum throughputs is still a factor of 100.

Figure 8 presents simulations results for the networks listed in Figure 7. The differences in traffic capacity can be clearly seen. Under the constant link width constraint (Figure 8(a)), the hypercube is clearly superior. The low-dimensional networks provide much higher latencies and only a small fraction of the capacity. When the node size is constrained (Figure 8(b)), the 10-dimensional network provides superior performance, followed by the 20-dimensional network. The 10-dimensional network provides slightly higher throughput even though the 20-dimensional network has a higher maximum throughput. As before, this is due to the larger link width allowing for better link utilization. Under the constant bisection width constraint (Figure 8(c)), the 4-dimensional network provides superior performance. The 10- and 20-dimensional networks suffer too greatly from the decreased link width and increased decode delays.

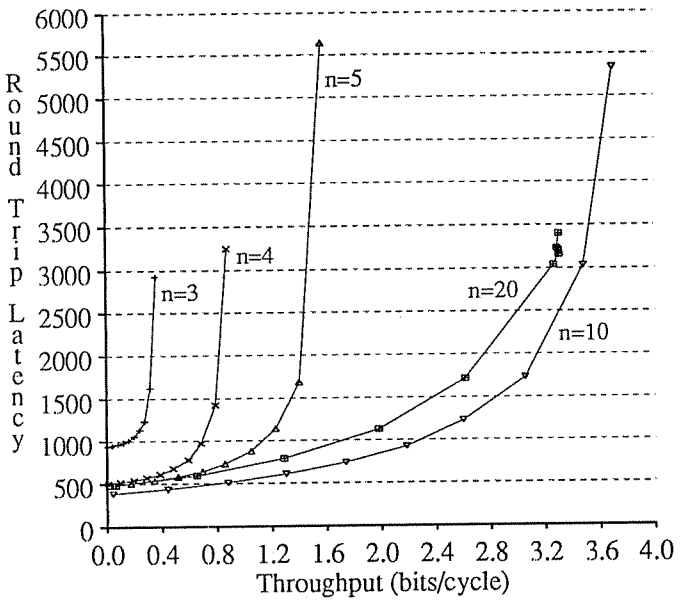
In Figure 9, configurations for a *synchronous*, 4096-node network are shown. Wire delay is now given as "Cycle Time Increase", the factor by which it increases the cycle time. The unloaded latency and maximum throughput are in terms of the base cycle time (without wire delay), to allow direct comparison with figures 5 through 8. The hypercube no longer gives the maximum throughput under the constant node size and bisection width constraints. This is due to the increase in cycle time for the higher-dimensional networks. The unloaded latency also displays a different relationship to the dimensionality, becoming much larger for higher-dimensional networks than was the case for pipelined networks.

Figure 10 presents simulation results for the networks listed in Figure 9. The results are noticeably different than those in Figure 6. Under the constant link width constraint (Figure 10(a)), the 6-dimensional network now provides superior performance. The 3- and 4-dimensional networks both provide *slightly* lower unloaded latency, but the 6-dimensional network clearly performs better as network traffic increases. Under the constant node size constraint (Figure 10(b)), the 3- and 4-dimensional networks are now best. The 12-dimensional network is now the worst choice. Finally, under the constant bisection width constraint (Figure 10(c)), the 3-dimensional network is clearly the best. The 6- and 12-dimensional networks provide very poor performance due to the greater wire delay and smaller link widths.

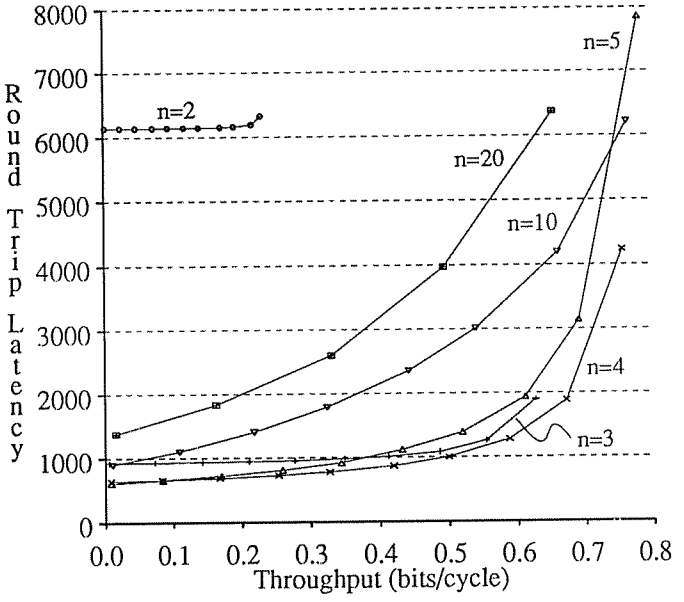
Several conclusions can be drawn from the data presented in Section 3. First, it is important to consider bandwidth as well as latency. The unloaded latency does not characterize the performance of a network as traffic



(a) Constant Link Width



(b) Constant Node Size



(c) Constant Bisection Width

Figure 8: Latency versus Throughput (Pipelined, N=1048576)

Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Cycle Time Increase	Decode Delay Cycles	Roundtrip Unloaded Latency	Maximum Throughput
2	64	32	128	4096	1.50	1	422.9	0.558
3	16	32	192	16384	1.50	1	182.4	2.342
4	8	32	256	32768	2.00	1	178.0	3.765
6	4	32	384	65536	3.00	1	213.0	5.856
12	2	32	768	131072	5.00	1	310.0	10.541

(a) Constant Link Width

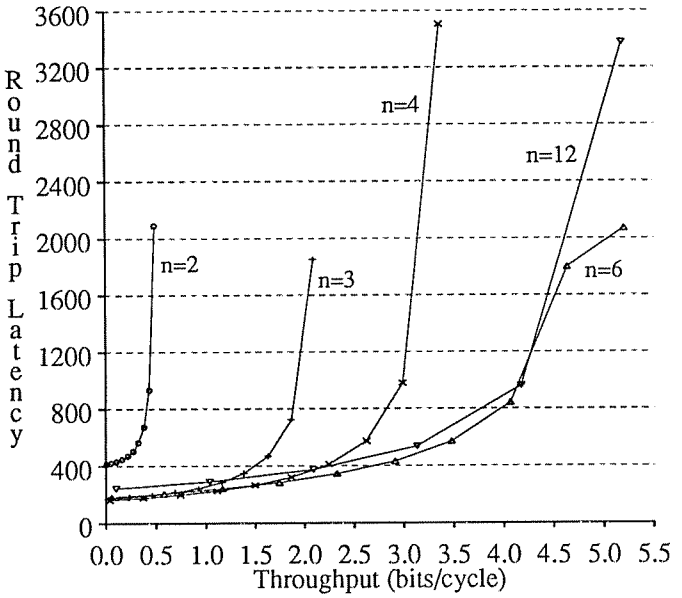
Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Cycle Time Increase	Decode Delay Cycles	Roundtrip Unloaded Latency	Maximum Throughput
2	64	48	192	6144	1.50	1	412.4	0.729
3	16	32	192	16384	1.50	1	182.4	2.342
4	8	24	192	24576	2.00	1	196.0	2.667
6	4	16	192	32768	3.00	1	285.0	2.928
12	2	8	192	32768	5.00	2	730.0	2.635

(b) Constant Node Size

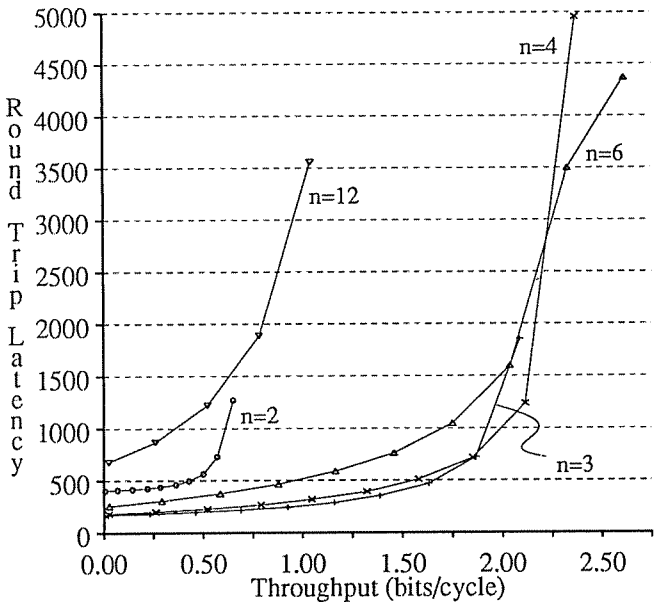
Dim	Radix	Link Width	Wires per Node	Wires across Bisection	Cycle Time Increase	Decode Delay Cycles	Roundtrip Unloaded Latency	Maximum Throughput
2	64	128	512	16384	1.50	1	395.9	1.896
3	16	32	192	16384	1.50	1	182.4	2.342
4	8	16	128	16384	2.00	1	226.0	1.882
6	4	8	96	16384	3.00	2	483.0	1.464
12	2	4	96	16384	5.00	3	1270.0	1.318

(c) Constant Bisection Width

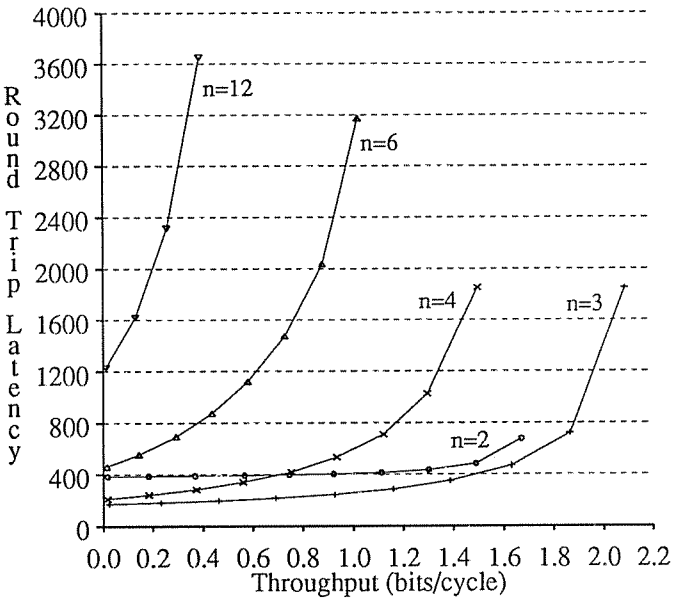
Figure 9: Actual System Configurations (Synchronous, N=4096)



(a) Constant Link Width



(b) Constant Node Size



(c) Constant Bisection Width

Figure 10: Latency versus Throughput (Synchronous, N=4096)

increases. Second, pipelined networks favor higher dimensionality than synchronous networks do. Pipelining allows the maximum wire length to grow without increasing the cycle time or causing the transmission delay across all links to grow. Finally, pipelining allows for much higher throughput than with synchronous networks.

This is illustrated by comparing figures 6 and 10. The throughputs attainable in the pipelined networks are considerably higher than those in the synchronous networks.

4. OTHER FACTORS IN NETWORK PERFORMANCE

This section explores several other factors in the performance of pipelined networks. Sections 4.1 and 4.2 investigate the effects of varying switching overhead and message length, respectively. The results differ somewhat from results obtained previously for synchronous networks [Agar91]. Section 4.3 investigates the effect of varying the queue and buffer sizes in a pipelined network switch.

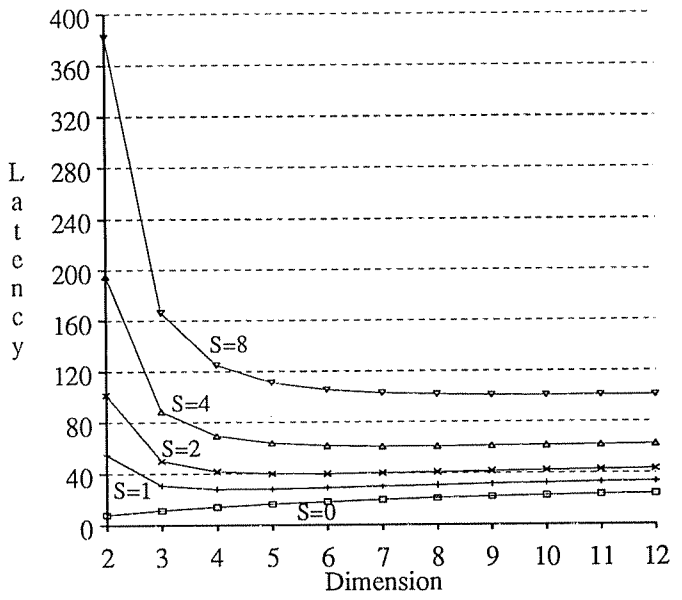
4.1. Effect of Switching Overhead

Figure 11 shows the unloaded latency versus dimensionality for a pipelined, 4096-node network as the ratio of switch cycle time to base wire delay, S, is varied from 0 to 8. Latencies are normalized to the cycle time with S=2. As in Figure 4, discrete quantities are treated as continuous to better illustrate relationships. In parts (a) and (b), the maximum wire length penalty is imposed on all links. In parts (c) and (d), mean (actual) wire lengths are used. Parts (a) and (c) assume the constant link width constraint, while parts (b) and (d) assume the constant node size constraint. The network with latency calculated using maximum wire length is presented only to aid intuition concerning wire-length effects.

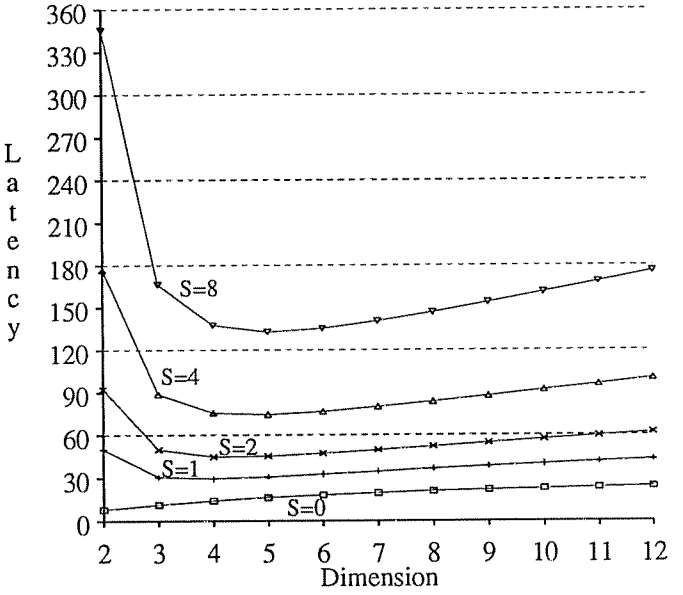
The optimal dimensionality is determined by balancing three components of latency that are affected by the dimensionality. (1) The mean number of hops is always reduced as dimensionality is increased. (2) Under the constant node size constraint, the required number of flits increases as dimensionality is increased. (3) Using the maximum wire penalty, wire delay is increased as dimensionality is increased. The delay due to the first two factors is proportional to the switch cycle time, while the delay due to the third factor is dependent on wire transmission time. Varying the ratio of switch cycle time to wire delay, therefore, affects the choice of dimensionality when the first or second factor is being balanced against the third factor. This is the case in figures 11(a) and (b). Increasing S makes the delay due to switching overhead more important relative to wire delay, and thus the optimal dimensionality increases. This agrees with previous results for synchronous networks [Agar91].

In figures 11(c) and (d), the mean (actual) wire length is used. As was explained in Section 2.1.2, the total wire delay in a pipelined network is roughly independent of dimensionality. Therefore, the optimal dimensionality for a given network size is determined solely by the link width and number of hops. The ratio of cycle time to base wire delay, S, *does not affect the optimal dimensionality*. It does affect the degree to which the dimensionality affects the latency, however. For small S, the impact of dimensionality on unloaded latency is small. For large S, the impact is greater.

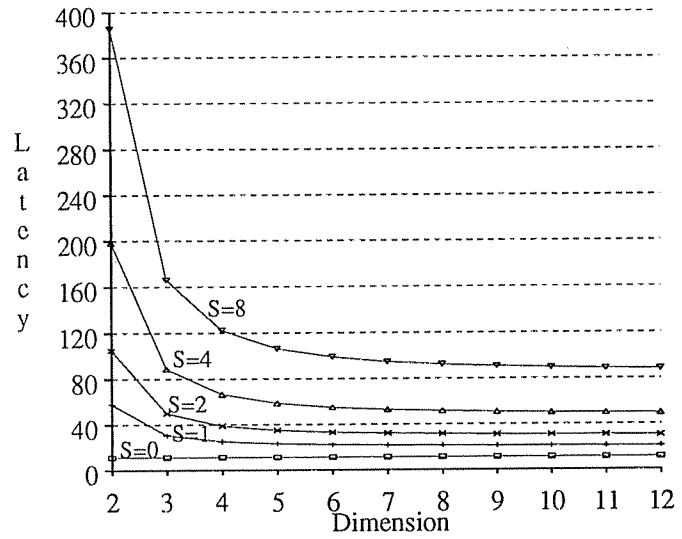
As switching speeds continue to increase, the value of S is likely to become smaller. In this case, the impact of dimensionality on the unloaded latency of pipelined networks will be reduced. However, bandwidth considerations will still favor the use of higher-dimensional networks. The advantage of pipelined networks will become



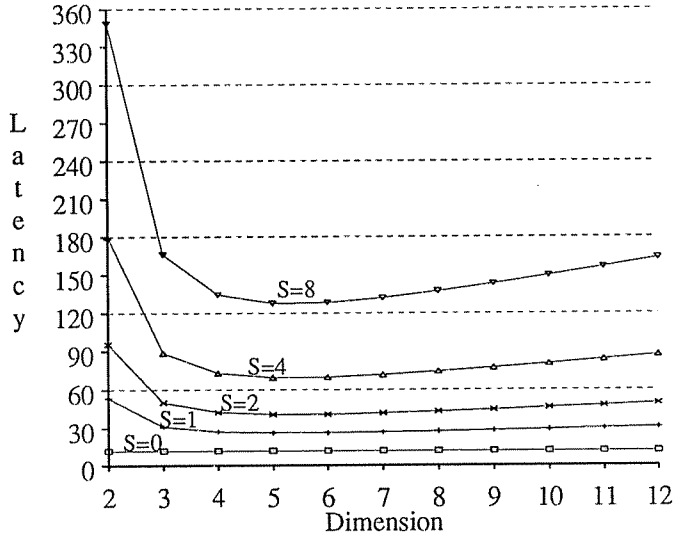
(a) Constant Link Width (Max Wire)



(b) Constant Node Size (Max Wire)



(c) Constant Link Width (Mean Wire)



(d) Constant Node Size (Mean Wire)

Figure 11: Effect of Switch to Wire Time Ratio
(N=4096, Pipelined Network)

greater as the switching rate grows in relation to the inverse of the wire latency. The optimal dimensionality of synchronous networks, which are greatly affected by wire latency, will shrink as the fraction of latency due to wire delay increases. This will reduce traffic capacity, especially for very large networks.

4.2. Effect of Message Lengths

Figure 12 shows the unloaded latency versus dimensionality for a pipelined, 4096-node network as the message length, L, is varied from 64 to 1024 bits. In parts (a) and (b), the maximum wire length penalty is imposed on all links. In parts (c) and (d), mean (actual) wire lengths are used. Parts (a) and (c) assume the constant link width constraint, while parts (b) and (d) assume the constant node size constraint.

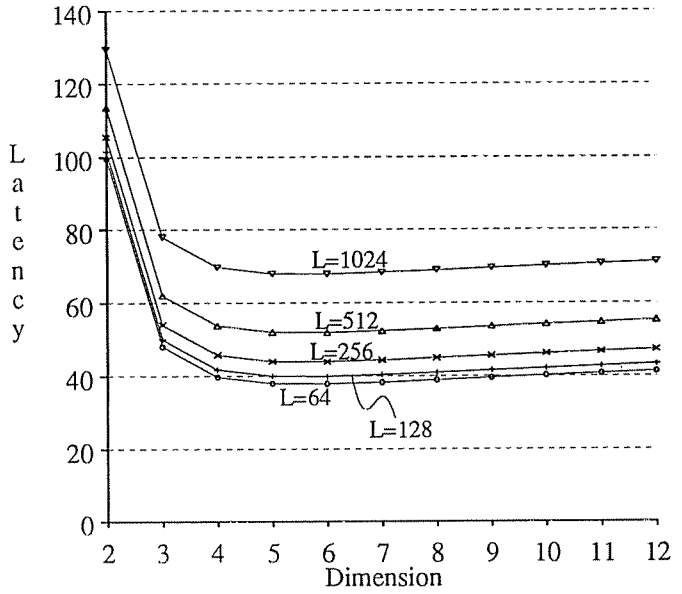
In synchronous networks, longer message lengths decrease the optimal dimensionality [Agar91]. This is because long messages lengths dominate the number of hops in the latency equation (8), making the reduction in number of hops in high-dimensional networks less important relative to the increased wire delay. A similar phenomenon can be seen in pipelined networks under the constant node size or bisection width constraints. Longer message lengths make the number of flits, P, a greater component of the latency (see equation 2), and the constraints favor low-dimensional networks, for which the link width is larger. This can be observed in figures 12(b) and (d). Under the constant node size constraint, using mean wire lengths (Figure 12(d)), the optimal dimensionality changes from 3, when L=1024, to 6, when L=64.

Under the constant link width constraint, however, the message length does *not* affect the optimal dimensionality. In this case, P still becomes a greater fraction of the latency for long messages (see equation 2), but since P is independent of dimensionality, it doesn't change the optimal dimensionality. This can be observed in figures 12(a) and (c), where L has no affect on the shape of the curves. The reason that pipelined networks behave differently than synchronous networks in this respect is due to the interaction between the number of hops, the number of flits and the wire lengths, as detailed in equations (4) and (8). In a synchronous network, the number of flits and the number of hops are added before being being multiplied by the wire length. The number of flits thus affects the balance between the number of hops and the wire lengths. In a pipelined network, where wire length does not determine cycle time, the number of flits is added to the product of wire length and number of hops. Thus, it does not affect their balance.

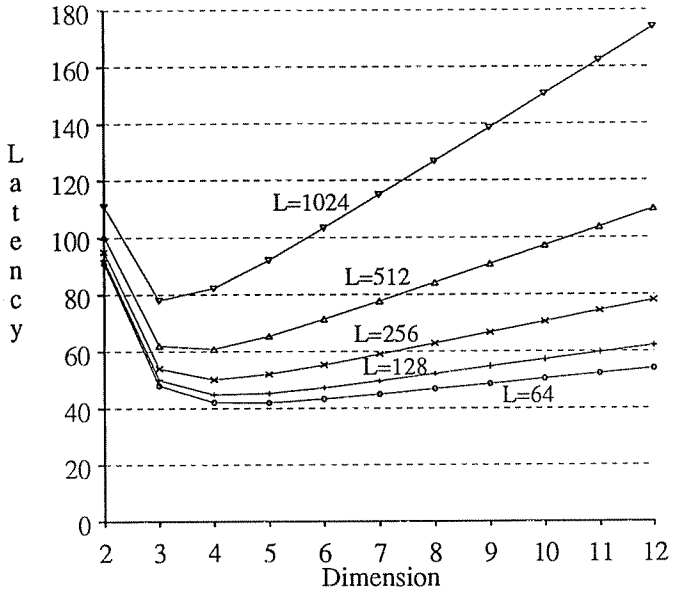
4.3. Effect of Queue Sizes

Figure 13 contains simulation results showing the effect of queue size on the performance of a pipelined network. The network size is 4096 (4 dimensional), the link width is 32 bits, and the latency is for a round trip message with 16 and 80 byte packets, as in Section 3. Queue sizes are given in flits. Except where noted, the ring buffer, input queue and output queue are all 100 bytes (25 flits).

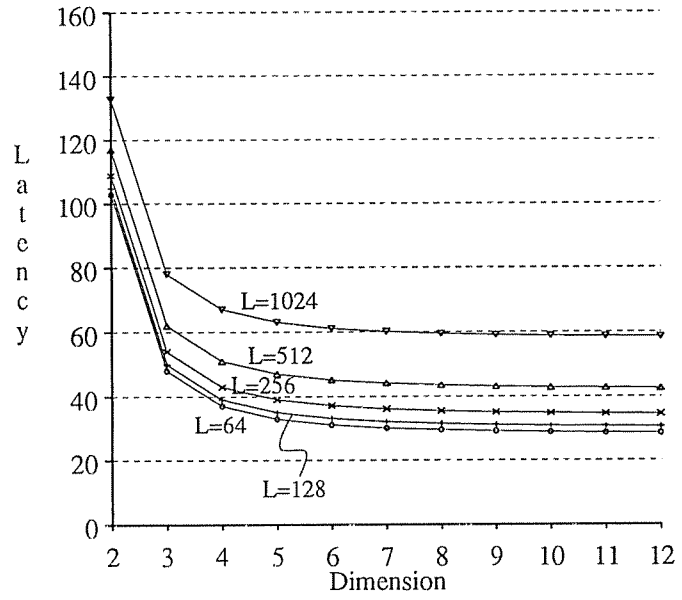
The ring buffer must be at least as large as the largest message (80 bytes) in order to guarantee that it will never fill up while the output queue is transmitting a message onto the output link. In Figure 13(a), the size of the ring buffer in flits is varied from 20 to ∞. The network is surprisingly insensitive to the buffer size. When the network is heavily loaded, it behaves slightly better with larger buffers, but the behavior is independent of buffer size throughout the normal operating region of the network. An actual implementation, therefore, should not spend extra resources making the ring buffers large.



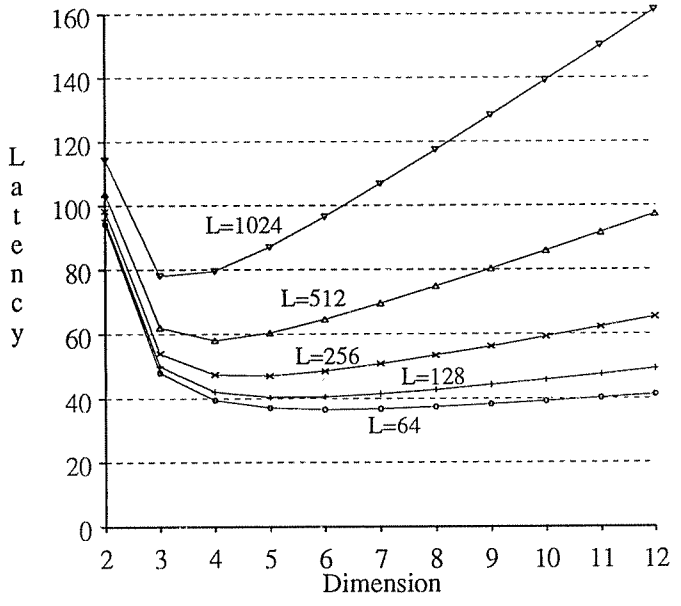
(a) Constant Link Width (Max Wire)



(b) Constant Node Size (Max Wire)

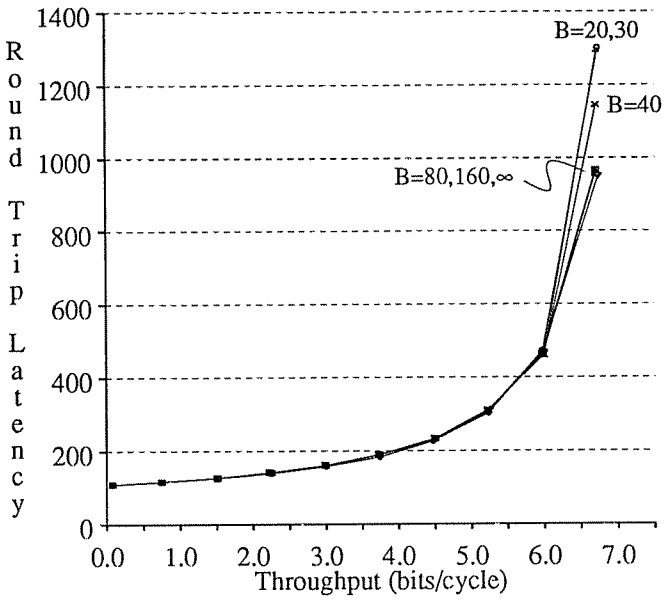


(c) Constant Link Width (Mean Wire)

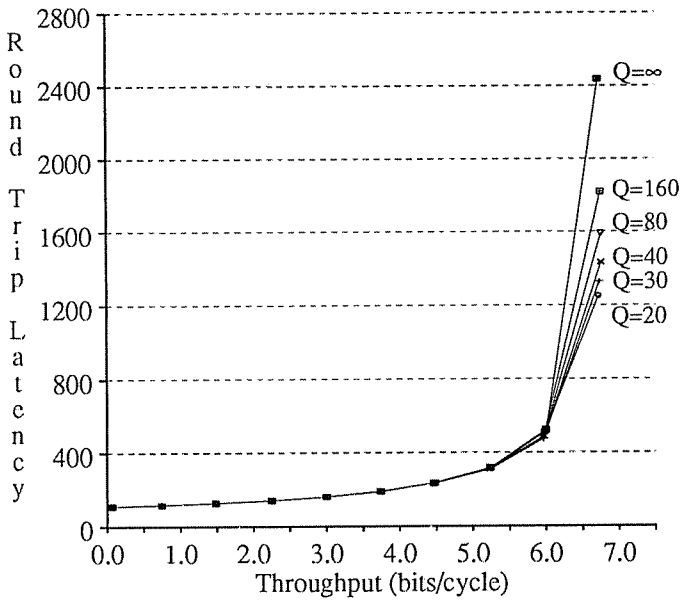


(d) Constant Node Size (Mean Wire)

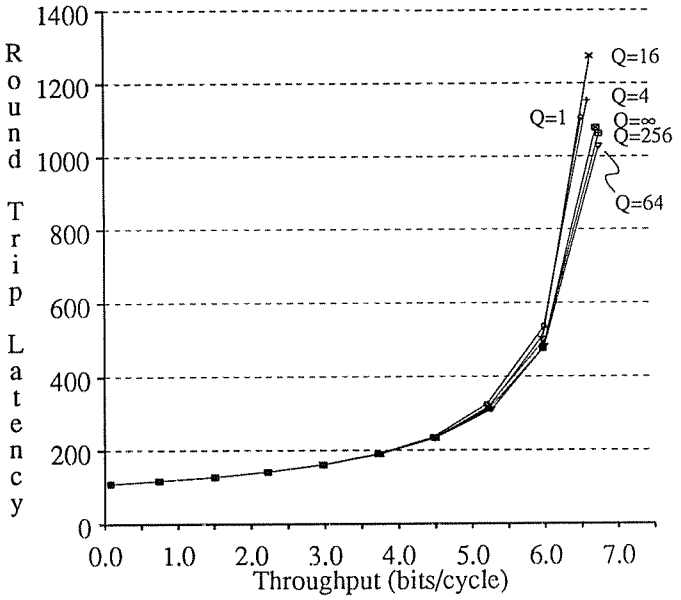
Figure 12: Effect of Message Length
(N=4096, Pipelined Network)



(a) FIFO Ring Buffer



(b) Input Queue



(c) Output Queue

Figure 13: Effect of Queue Sizes
(N=4096, Pipelined Network)

The input queue also must be large enough to hold the longest message, as its free space is checked when a message arrives in order to determine if the message can be accepted. An alternate design could accept an

incoming message one flit at a time, and send a negative acknowledgement only if the queue fills up completely. At the time the queue runs out of space, the switch could have already started sending the acknowledgement, in which case it must be able to *stomp* the end of the acknowledgement, turning it into a negative acknowledgement. This scheme is used in the SCI [IEEE90]. In Figure 13(b), the input queue size in flits is varied from 20 to ∞. As with the ring buffer, the network is quite insensitive to the input queue size. Performance is identical for all buffer sizes throughout the network's normal operating region. Under heavy loading, larger queue sizes actually increase the latency. The packet drop rate is higher for the smaller queues, but the waiting time in each queue is substantially lower. The input queue, therefore, should not be made larger than necessary.

The output queues do not have to be large enough to hold an entire message. If a message is partially accepted into an output queue and the queue runs out of space, the message is simply backed up in the switch, preventing the connected input queue from draining. In Figure 13(c), the output queue size is varied from 1 to ∞ flits. As with the ring buffer and input queue, the output queue does not affect performance until the network is heavily loaded. The latency under heavy loading rises somewhat as the queue size is increased from 1 to 4 to 16 flits (for the same reason the latency rises as the input queue size is increased), and then drops when the queue size is set to 64 flits. Further experiments have indicated that the drop in latency occurs when the output queue is large enough to hold an entire data packet. This reduces the probability that a message is blocked while "straddling" the switch. Depending upon the implementation costs, it may be worthwhile to make the output queues large enough to hold the largest message.

5. CONCLUSIONS

This paper has examined the performance of pipelined, k-ary n-cube networks. The key attribute of pipelined networks is that, by allowing multiple bits to be in flight on the same wire, the throughput of a link is divorced from the latency. The switch cycle time is independent of wire length, and can be kept small, as it does not include transmission time. This removes the primary disadvantage of high dimensionality for large networks, that of increased latency and decreased throughput due to long wires. By doing so, the network design tradeoffs are significantly changed.

The optimal dimensionality of pipelined networks is higher than that of synchronous networks. In a synchronous network, wire delays discourage the increase of dimensionality as system size is increased. This has the effect of making the optimal radix increase with system size. In a pipelined network, wire delays have little affect on the choice of dimensionality. The optimal radix remains roughly constant as system size is increased. When the link width is unconstrained, a hypercube (radix of 2) always provides the best performance. Under the constant node size constraint, the optimal radix is between 5 and 10. Since the dimensionality and radix of a network must be whole numbers, actual design choices will be limited. A 4096-node network, for example, can have a radix of 2, 4, 8, 16 or 64. Under the constant node size constraint, the best choice is 4 (6-dimensional).

It is important to consider throughput as well as latency, as the two are intimately related. Since the traffic density per link is proportional to the network radix, high-radix networks have a correspondingly lower traffic capacity. Under the constant node size constraint this is offset by their larger link widths, but the capacity still decreases as radix increases. In a synchronous network, the optimal radix increases with system size, causing the per processor traffic capacity to decrease. If the radix is kept small, then the high dimensionality causes the maximum wire length to grow, thus increasing the cycle time and decreasing throughput. In a pipelined network, the optimal radix, and thus the per processor traffic capacity, remain constant as system size grows.

Pipelined networks offer higher throughput and lower latency than do synchronous networks. The advantages of pipelined networks will become greater as system sizes increase, and as switching times become smaller relative to transmission delays.

References

[Abra89] Abraham, S. and K. Padmanabhan, Performance of the Direct Binary n-Cube Network for Multiprocessors, *IEEE Transactions on Computers* **38**(7), July 1989, 775-785.

[Adve91] Adve, V. S. and M. K. Vernon, Performance Analysis of Multiprocessor Mesh Interconnection Networks with Wormhole Routing, Computer Sciences Technical Report #1001, University of Wisconsin-Madison, Madison, WI 53706, February 1991.

[Agar91] Agarwal, A., Limits on Network Performance, *IEEE Transactions on Parallel and Distributed Systems (to appear)*, 1991.

[Alve90] Alverson, R., D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith, The Tera Computer System, *Proc. 1990 International Conference on Supercomputing*, June 1990.

[Bork88] Borkar, S., R. Cohn, G. Cox, S. Gleason, T. Gross, H. T. Kung, M. Lam, B. Moore, C. Peterson, J. Pieper, L. Rankin, P. S. Tseng, J. Sutton, J. Urbanski, and J. Webb, iWarp: An Integrated Solution to High-Speed Parallel Computing, *Proc. Supercomputing '88*, November 1988.

[Dall90] Dally, W. J., Performance Analysis of k-ary n-cube Interconnection Networks, *IEEE Transactions on Computers* **39**(6), June 1990, 775-785.

[Dias83] Dias, D. M. and J. R. Jump, The Performance of Multistage Interconnection Networks for Multiprocessors, *IEEE Transactions on Computers* **C-32**(12), December 1983, 1091-1098.

[Feng81] Feng, T., A Survey of Interconnection Networks, *Computer*, December, 1981, 12-27.

[Hill85] Hillis, W. D., *The Connection Machine*, MIT Press, Cambridge, MA, 1985.

[IEEE90] IEEE,, Scalable Coherent Interface, Logical Specification, IEEE Standard Specification P1596: Part II, November 1990.

[Kerm79] Kermani, P. and L. Kleinrock, Virtual Cut-Through: A New Computer Communication Switching Technique, *Computer Networks* **3**, 1979, 267-286.

[Krus83] Kruskal, C. P. and M. Snir, The Performance of Multistage Interconnection Networks for Multiprocessors, *IEEE Transactions on Computers* **C-32**(12), December 1983, 1091-1098.

[Lawr75] Lawrie, D. H., Access and Alignment of Data in an Array Processor, *IEEE Transactions on Computers* **C-24**(12), December 1975, 1145-1155.

[Lee84] Lee, M. and C. Wu, Performance Analysis of Circuit Switching Baseline Interconnection Networks, *Proc. 11th Annual International Symposium on Computer Architecture*, June 1984, 82-90.

[Leno89] Lenoski, D., J. Laudon, K. Gharachorloo, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, Design of the Stanford DASH Multiprocessor, CSL Technical Report 89-403, Stanford University, December 1989.

[Pate81] Patel, J. H., Performance of Processor-Memory Interconnections for Multiprocessors, *IEEE Transactions on Computers* **C-30**(10), October 1981, 771-780.

[Scot91] Scott, S. L., A Cache Coherence Mechanism for Scalable, Shared Memory Multiprocessors, *Proc. 1991 International Symposium on Shared Memory Multiprocessing*, April 1991.

[Seit84] Seitz, C. L., Concurrent VLSI Architectures, *IEEE Transactions on Computers* **33**(12), December 1984, 775-785.

[Seit85] Seitz, C. L., The Cosmic Cube, *Communications of the ACM* **28**(1), January 1985, 22-33.

[Sieg79] Siegel, H. J., Interconnection Networks for SIMD Machines, *Computer* **12**(6), June, 1979, 57-65.

[Sull77] Sullivan, H. and T. R. Bashkow, A Large Scale, Homogeneous, Fully Distributed Parallel Machine, *Proc. 4th Annual International Symposium on Computer Architecture*, March 1977, 105-117.

[Yoon87] Yoon, H., K. Y. Lee, and M. T. Liu, Performance Analysis and Comparison of Packet Switching Interconnection Networks, *Proc. 1987 International Conference on Parallel Processing*, August 1987, 542-545.